SQL INJECTION VULNERABILITY

What is SQL Injection ?

- SQL injection is a type of security vulnerability that occurs when an attacker injects malicious code into an SQL statement. This is done by inserting code into a web form input field or a URL parameter that is then passed to the database.

- If the application does not properly sanitize or validate the user input, the injected SQL code can be executed by the database, allowing the attacker to access, modify, or delete sensitive data.

- For example, an attacker could use SQL injection to bypass authentication and gain access to an application or database. They could also steal sensitive data such as usernames, passwords, and credit card information.

- Preventing SQL injection involves validating and sanitizing user input, using prepared statements or parameterized queries, and restricting database privileges. It is important for developers to be aware of this vulnerability and take steps to secure their applications against it.

Impact of SQL Injection

- SQL injection attacks can have serious and far-reaching impacts on web applications, databases, and the businesses and individuals who use them. Here are some of the potential impacts of SQL injection attacks.

- Unauthorized access to sensitive data: SQL injection can allow attackers to gain unauthorized access to sensitive information, such as user credentials, credit card numbers, personal information, and other confidential data stored in the database.

- Data modification or deletion: Attackers can use SQL injection to modify or delete data in the database, which can result in data loss, corruption, or inaccuracy. This can lead to financial losses, reputation damage, or legal liabilities.

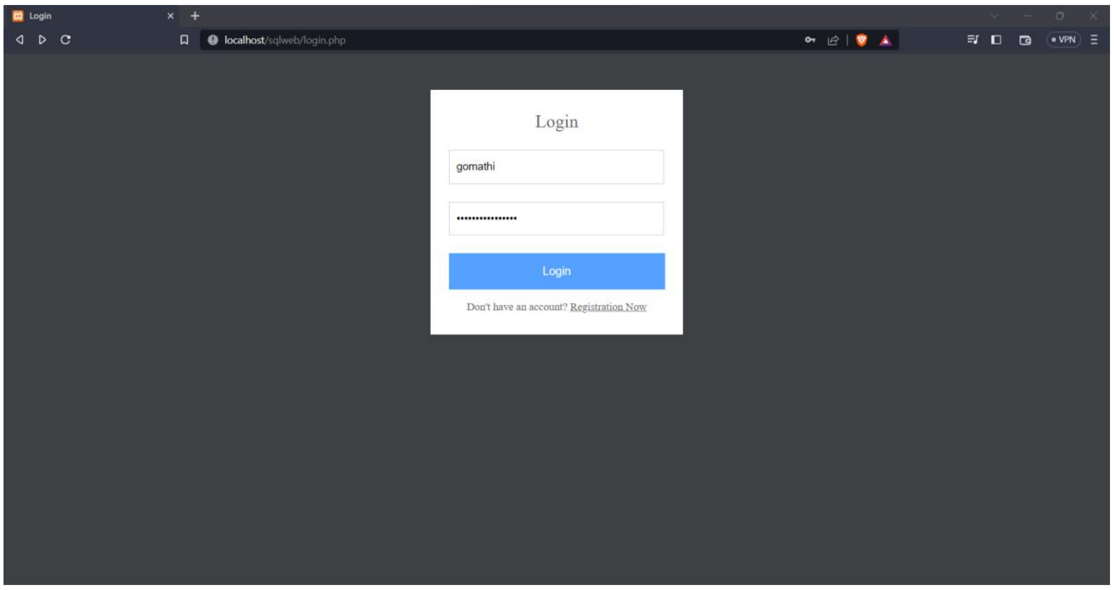
- Application and database downtime: SQL injection attacks can cause web applications and databases to crash, resulting in service downtime, loss of productivity, and revenue losses.

- Malware and other security threats: SQL injection can also be used as a delivery mechanism for malware and other types of security threats that can compromise the integrity and security of the entire system.

- Legal and regulatory consequences: SQL injection attacks can result in legal and regulatory consequences, including fines, lawsuits, and compliance violations.

SQL Injection Authentication Bypass using simple SQL Injection Commands

Username : Gomathi
Password: 1’ or 2=2 --true



Login

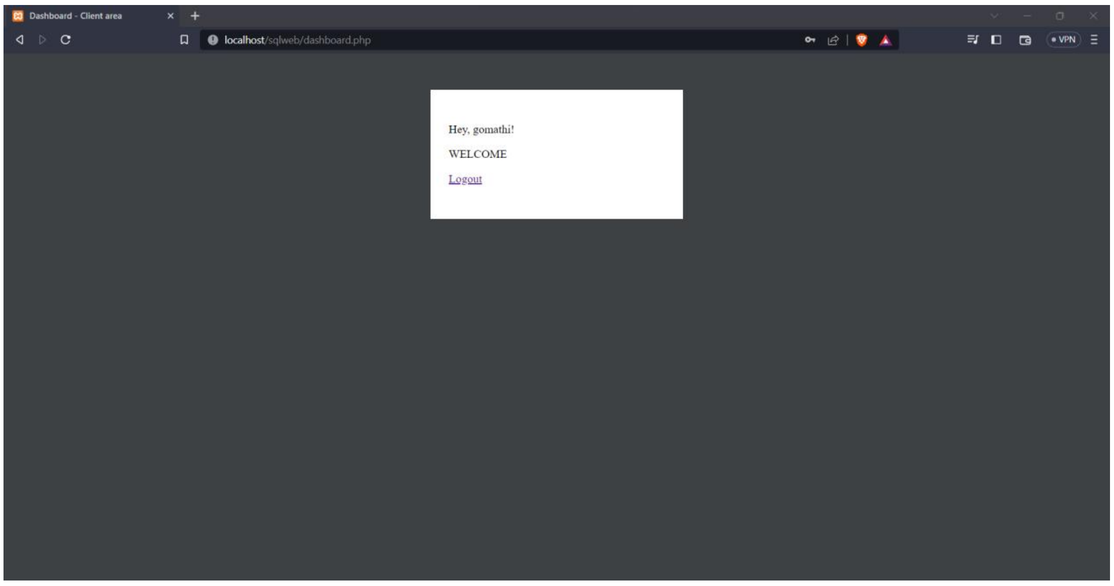
gomathi

••••••••••••••••

Login

Don't have an account? Registration Now

OUTPUT



Hey, gomathi!

WELCOME

Logout

How to prevent SQL Injection Vulnerability?

- Parameterized queries: Use parameterized queries or prepared statements to dynamically pass user inputs to the database, instead of concatenating user input with SQL queries. This can prevent malicious SQL code injection by properly escaping special characters in user input.

- Input validation and sanitization: Validate and sanitize user input by checking for the expected data type, length, and format, and filtering out any special characters that may be used in SQL injection attacks.

- Least privilege principle: Grant database users the minimum level of privileges necessary to perform their intended functions. This can limit the potential impact of a SQL injection attack by preventing attackers from accessing sensitive information or modifying data.

- Error messages: Do not display error messages that disclose information about the database or the application's underlying architecture. This can help prevent attackers from exploiting SQL injection vulnerabilities.

- Regular security assessments: Regularly conduct security assessments and penetration testing on your web applications and databases to identify and mitigate potential SQL injection vulnerabilities.

- Keep your software up-to-date: Keep your web application and database software up-to-date to ensure that any known vulnerabilities are patched and up-to-date security measures are implemented.

- User access control: Limit access to your web application and database by implementing user access control, authentication and authorization.

CROSS SITE SCRIPTING VULNERABILITY (XSS)

What is Cross site scripting ?

- In a cross-site vulnerability attack, an attacker injects malicious code into a website that is viewed by other users. The code can then be executed in the users' web browsers, potentially allowing the attacker to steal sensitive information or perform other malicious actions on the users' behalf.

- There are several types of cross-site vulnerabilities, including stored, reflected, and DOM-based XSS. Stored XSS occurs when the malicious code is permanently stored on the server and served to all users who access the affected page, while reflected XSS occurs when the code is included in the URL or other input and reflected back to the user's browser. DOM-based XSS occurs when the code is executed in the Document Object Model (DOM) of the web page, rather than in the server's response.

Impact of Cross Site Scripting :

- ❖ Data theft: Attackers can use cross-site vulnerabilities to steal sensitive information, such as login credentials, credit card numbers, or personal data, from users who visit the affected web page.

- ❖ Malware distribution: Attackers can use cross-site vulnerabilities to distribute malware, such as viruses or trojans, to users who visit the affected web page. This can result in the infection of the user's device and compromise their security.

- ❖ Website defacement: Attackers can use cross-site vulnerabilities to modify the content of the affected web page, potentially damaging the reputation of the website or its owner.

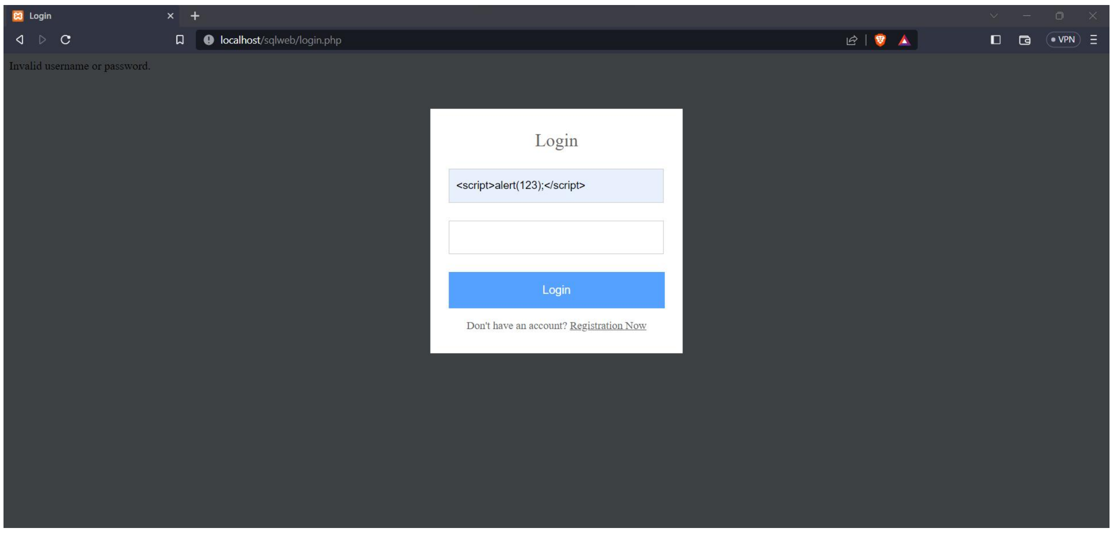
- ❖ Reputation damage: If a website is found to be vulnerable to cross-site attacks, it can damage the trust and reputation of the website owner and their business.

- ❖ Legal and regulatory consequences: Cross-site vulnerabilities can violate data protection laws and regulations, resulting in legal and regulatory consequences, such as fines or legal action.

- ❖ Financial losses: Cross-site vulnerabilities can result in financial losses for the website owner or users, such as loss of revenue, lawsuits, or identity theft

CROSS SITE SCRIPTING VULNERABILITY

Example script : <script>alert(123);</script>



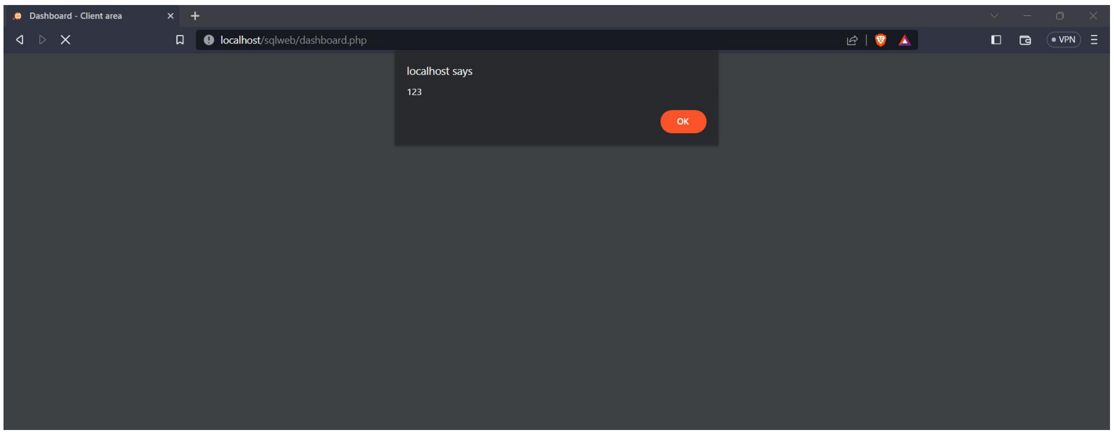
Invalid username or password.

Login

<script>alert(123);</script>

Login
localhost/sqlweb/login.php
Login
Don't have an account? Registration Now

•



localhost says

123

How to prevent from Cross Site Scripting Vulnerability ?

❖ Sanitize User Input: Validate and sanitize all user input, including form data, cookies, and HTTP headers. Use input validation to ensure that all input meets the expected format, and sanitize any user-supplied data before it is displayed on the page.

❖ Use Encoding and Filtering: Use encoding and filtering techniques to prevent malicious code injection. HTML entities, JavaScript escaping, and URL encoding can all be used to filter out or encode special characters that may be used in XSS attacks.

❖ Use Content Security Policy (CSP): Implement a content security policy (CSP) to specify which sources of content are allowed to be loaded by your website. This helps to prevent XSS attacks by blocking the loading of any resources from untrusted domains.

❖ Use HTTPOnly Cookies: Set the HTTPOnly flag on all cookies to prevent client-side script from accessing them. This makes it more difficult for attackers to steal session tokens and other sensitive information.

❖ Keep Your Software Up-to-Date: Keep all software up-to-date, including web servers, databases, and any third-party libraries or frameworks that you use. Patches and updates often include security fixes that address known vulnerabilities.

❖ Educate Users: Educate your users about the risks of XSS attacks and how to avoid them. This includes training users to be cautious when clicking on links or entering sensitive information into web forms.

SESSION HIJACKING VULNERABILITY

What is Session Hijacking ?

- Session hijacking is a type of security vulnerability that occurs when an attacker intercepts and takes over a user's active session on a website or web application. This allows the attacker to gain access to sensitive information, perform unauthorized actions, or impersonate the user.

- There are different methods for carrying out session hijacking attacks, but most involve stealing the user's session ID or token, which is used to authenticate and authorize the user's requests to the server. Once the attacker has obtained the session ID or token, they can use it to access the user's account or perform actions on their behalf.

- Session hijacking attacks can be particularly dangerous on websites that deal with sensitive information such as financial transactions or personal data. To prevent session hijacking, web developers can use techniques such as encryption, secure cookies, and session timeouts to make it more difficult for attackers to steal and use session IDs or tokens. Additionally, users should be cautious when using public Wi-Fi networks or accessing websites from untrusted devices, as these can increase the risk of session hijacking attacks.

Impact of Session Hijacking Vulnerability ?

- Theft of personal information: If a user's session is hijacked, the attacker may be able to access personal information such as login credentials, credit card numbers, and other sensitive data.

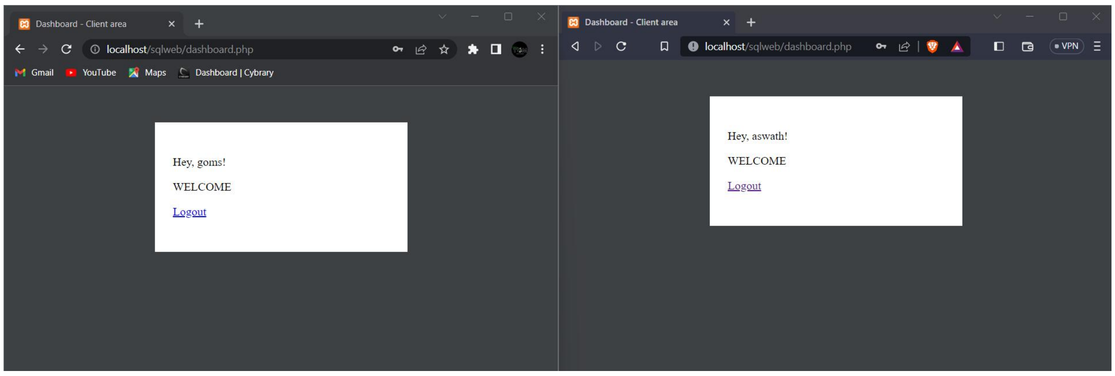
- Unauthorized access to accounts: With access to a user's session, the attacker may be able to log in to the user's account and perform actions such as making purchases, changing settings, or deleting data.

- Data breaches: If an attacker gains access to a session that has privileged access to a database or other sensitive resources, they may be able to cause a data breach that exposes sensitive information to unauthorized parties.

- Reputation damage: If a website or application is found to be vulnerable to session hijacking attacks, it can damage the organization's reputation and erode user trust.

- Legal and financial repercussions: In some cases, a session hijacking attack can lead to legal or financial repercussions for the organization responsible for the website or application, particularly if sensitive data is compromised or stolen.

SESSION HIJACKING VULNERABILITY



Hey, goms!

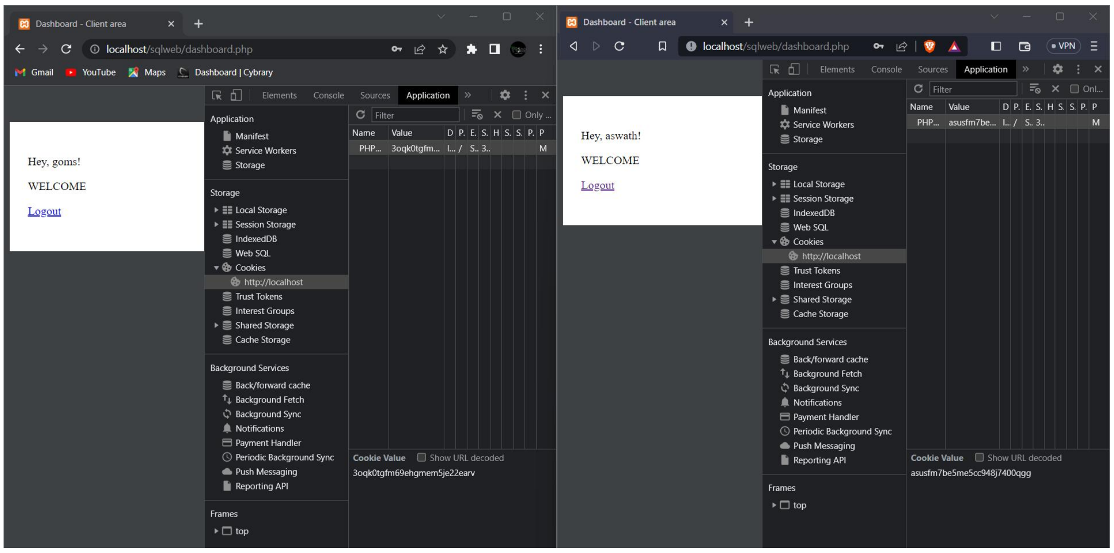
WELCOME

Logout

Hey, aswath!

WELCOME

Logout



Hey, goms!

WELCOME

Logout

Name	Value
PHP...	3oqk0tgfm...

Cookie Value

3oqk0tgfm69ehgmem5je22earv

Hey, aswath!

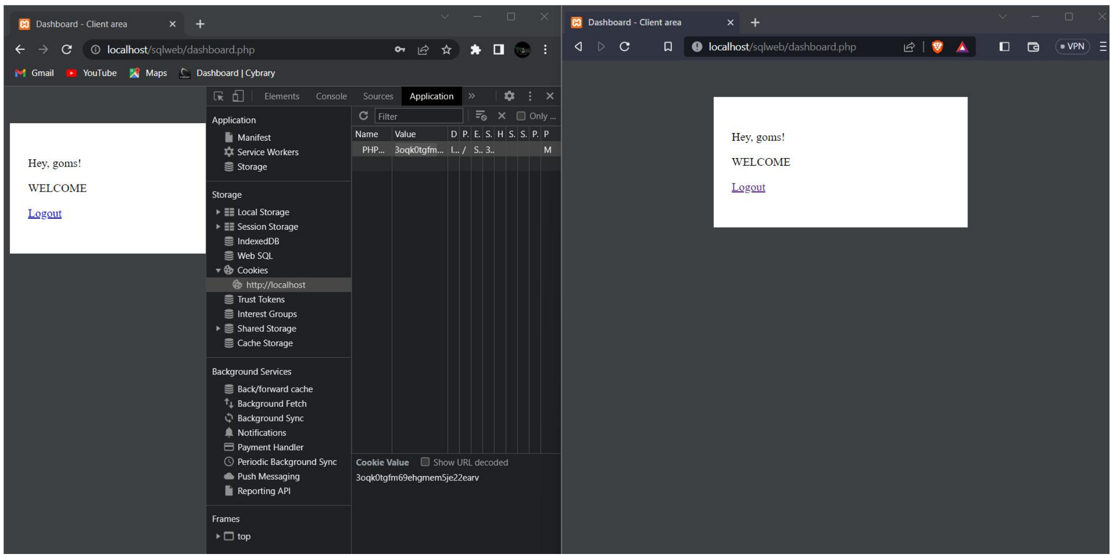
WELCOME

Logout

Name	Value
PHP...	asusfm7be...

Cookie Value

asusfm7be5me5cc948j7400qgg



Hey, goms!

WELCOME

Logout

Name	Value
PHP...	3oqk0tgfm...

Cookie Value

3oqk0tgfm69ehgmem5je22earv

Hey, goms!

WELCOME

Logout

How to prevent from Session Hijacking Vulnerability ?

- Use HTTPS: HTTPS encrypts all data sent between a web server and a client, making it more difficult for an attacker to intercept and steal session data.
- Use secure cookies: Secure cookies are encrypted and can only be transmitted over HTTPS. This makes it harder for an attacker to intercept and use a session cookie to hijack a user's session.
- Implement session timeouts: Set an expiration time for user sessions, forcing the user to re-authenticate after a period of inactivity.
- Use multi-factor authentication: Multi-factor authentication adds an extra layer of security by requiring users to provide additional information, such as a code sent to their phone or a fingerprint scan.
- Use anti-CSRF tokens: Anti-CSRF tokens can prevent cross-site request forgery attacks, which can be used to hijack a user's session.
- Regularly update software: Keep web servers, web applications, and third-party libraries up-to-date to reduce the risk of session hijacking vulnerabilities due to unpatched security flaws.

Train users on security best practices: Educate users on best practices for keeping their accounts secure, such as using strong passwords and avoiding public Wi-Fi networks.