Name:Ashwin kumar.R
Class:1st Msc cyber security
sub:Advance ethical hacking and penetration testing

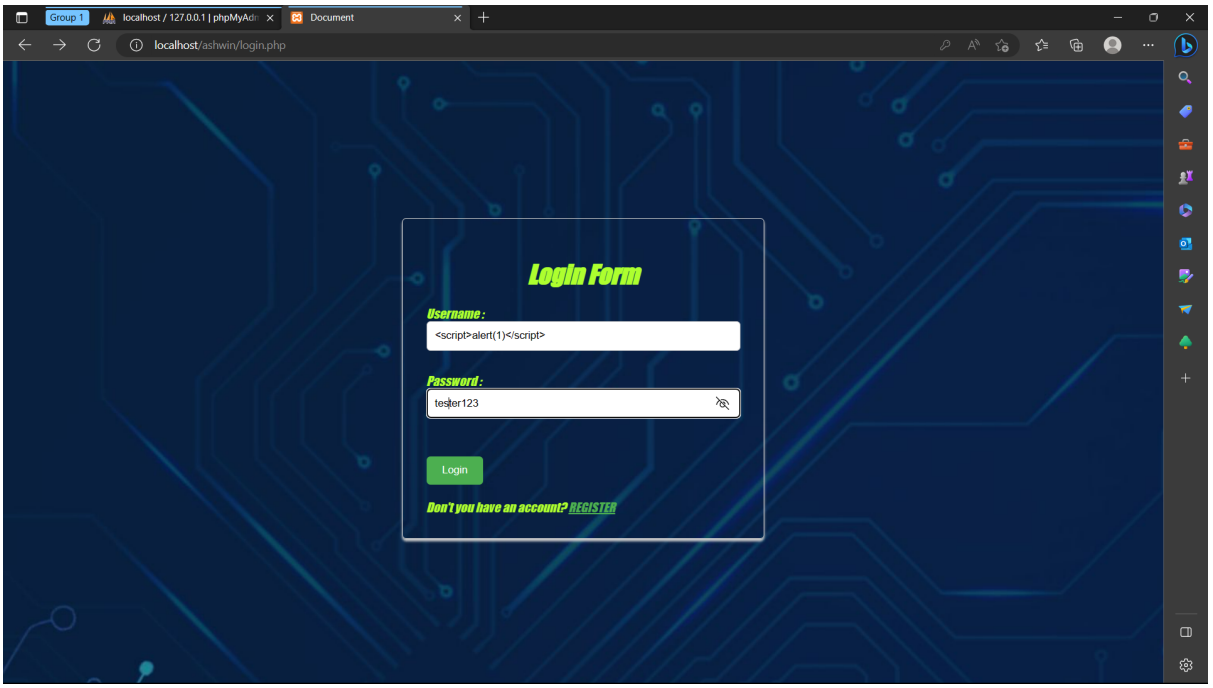
Cross-site scripting (XSS) vulnerabilities are a type of security vulnerability that allows an attacker to inject malicious scripts or code into web pages viewed by other users. XSS vulnerabilities typically arise when a website fails to properly sanitize user input, allowing an attacker to inject malicious code that can be executed by other users who visit the site.

There are several types of XSS vulnerabilities, including:

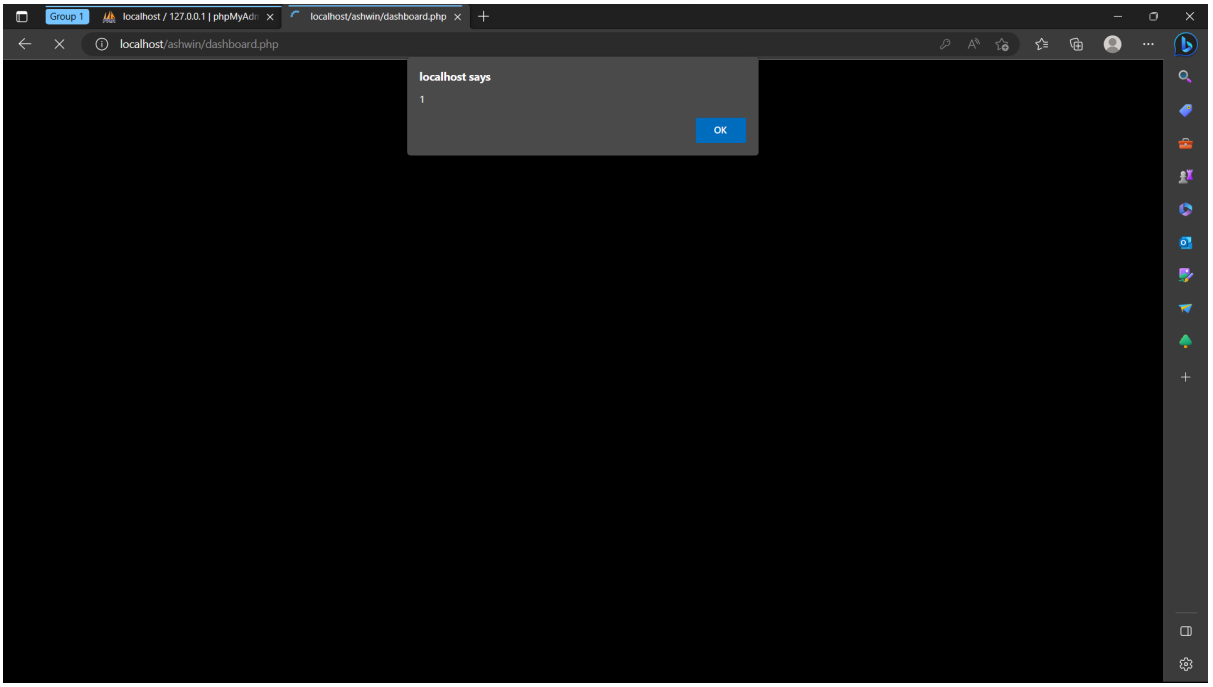
1. Reflected XSS: In this type of attack, the attacker sends a malicious script to the server, which then reflects the script back to the user's browser. The user's browser executes the script, allowing the attacker to steal sensitive information or perform other malicious actions.
2. Stored XSS: In this type of attack, the attacker injects a malicious script into a web page that is stored on the server. When other users view the page, the script is executed, allowing the attacker to steal sensitive information or perform other malicious actions.
3. DOM-based XSS: In this type of attack, the attacker injects a malicious script into the Document Object Model (DOM) of a web page. The script is then executed by the user's browser, allowing the attacker to steal sensitive information or perform other malicious actions.

Preventing XSS vulnerabilities requires careful input validation and sanitization. Web developers should ensure that all user input is properly sanitized before being displayed oweb page, and should use encoding techniques such as HTML entity encoding or URL encoding to prevent malicious scripts from being executed. Additionally, web developers can use Content Security Policy (CSP)

Cross Site Scripting Using This Payloads <script>alert(1)</script>



Output:



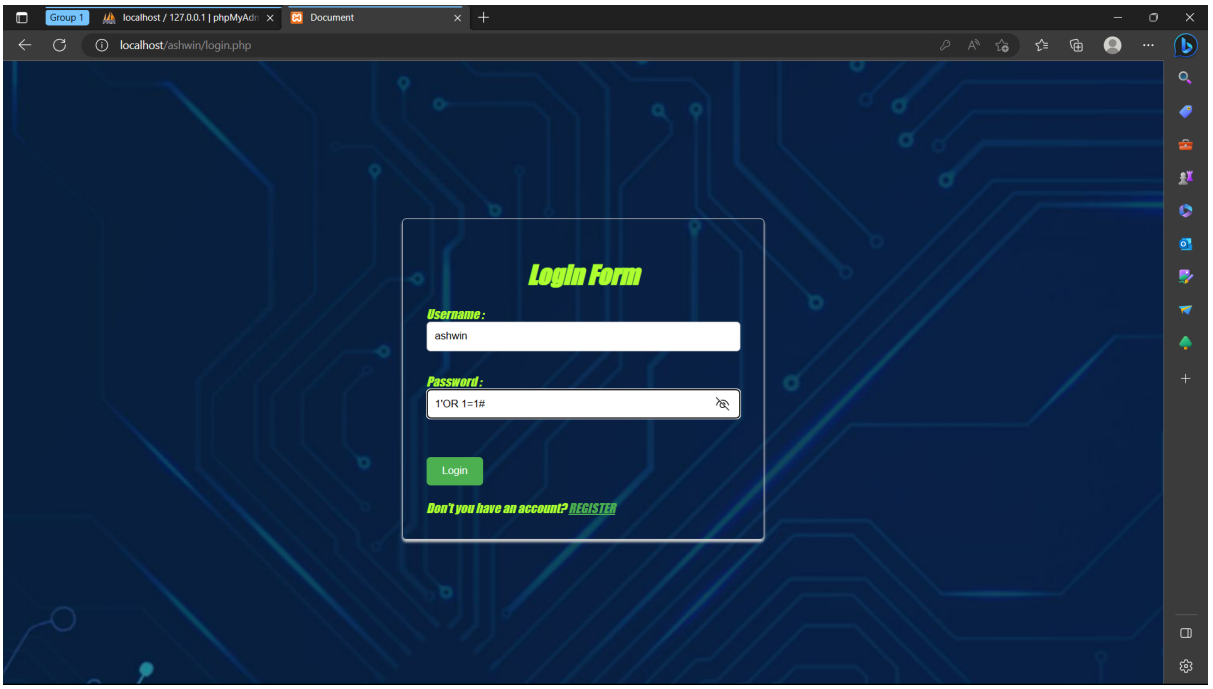
SQL INJECTIONS VULNERABILITIY

SQL injection is a type of security vulnerability that occurs when an attacker inserts malicious SQL code into a website's input fields, such as a login form or search bar, which then executes in the website's database. This can result in unauthorized access to sensitive data, modification of data, and even the complete destruction of the database.

SQL injection attacks can be prevented by using prepared statements and parameterized queries, which allow the input values to be treated as data instead of executable code. It is also important to sanitize and validate user input, and to limit database permissions to only those necessary for the application to function.

To protect against SQL injection, it is important to understand how it works and to use secure coding practices to prevent it from occurring. It is also important to regularly monitor and update your website's security measures to stay ahead of potential vulnerabilities.

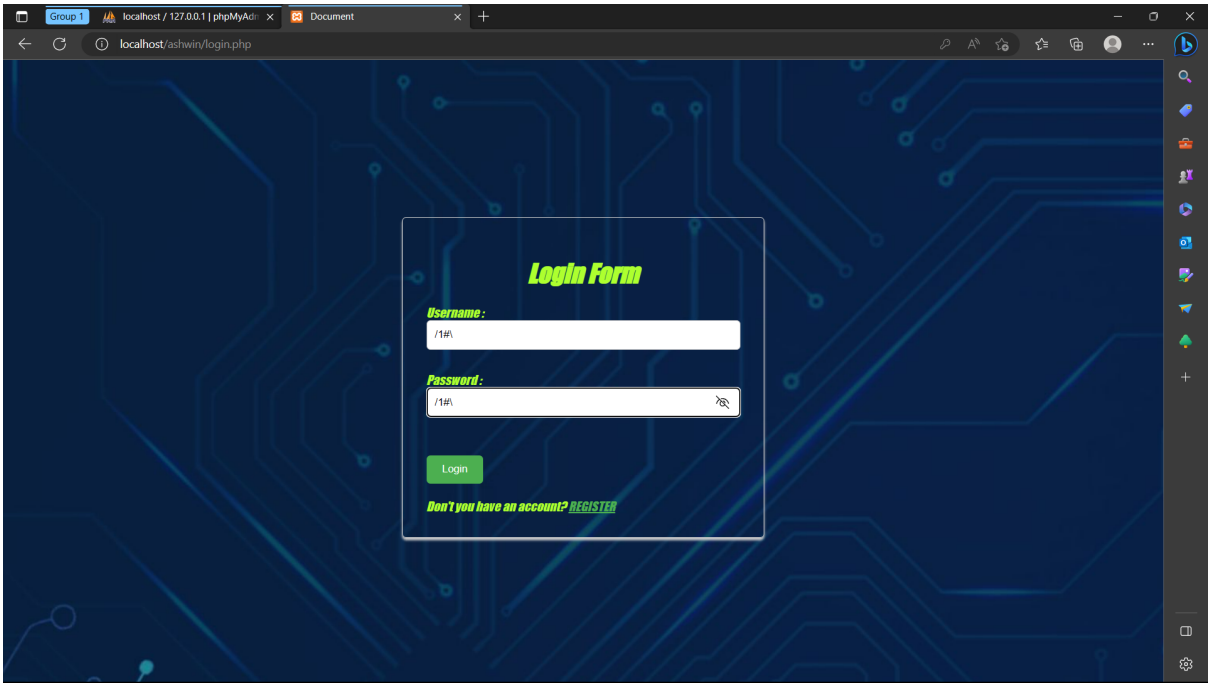
1.SQL Injections Authentication Bypass using SQL injection Commands:



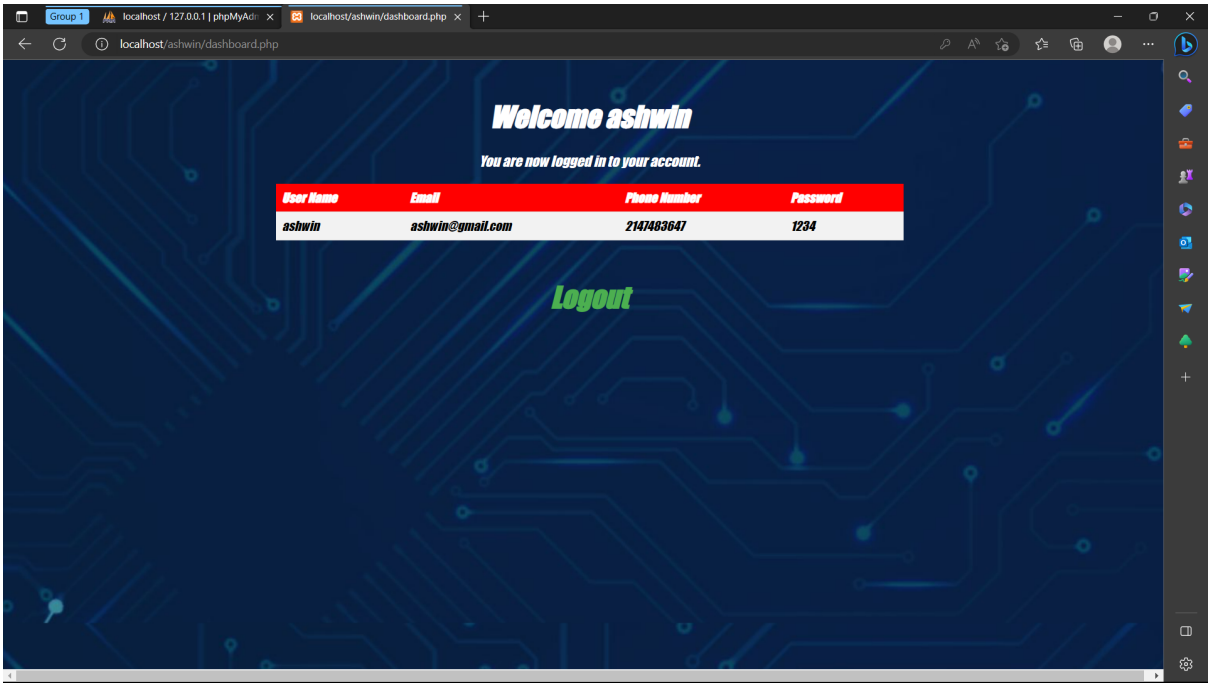
2.using another sql simple command bypassing the login authentication

username:/1#\

password:/1#\


Group 1
localhost / 127.0.0.1 | phpMyAdm
Document
localhost/ashwin/login.php
Login Form
Username :
/1#\
Password :
/1#\
Login
Don't you have an account? REGISTER

Output


Group 1
localhost / 127.0.0.1 | phpMyAdm
localhost/ashwin/dashboard.php
localhost/ashwin/dashboard.php
Welcome ashwin
You are now logged in to your account.
User Name
Email
Phone Number
Password
ashwin
ashwin@gmail.com
2147483647
1234
Logout

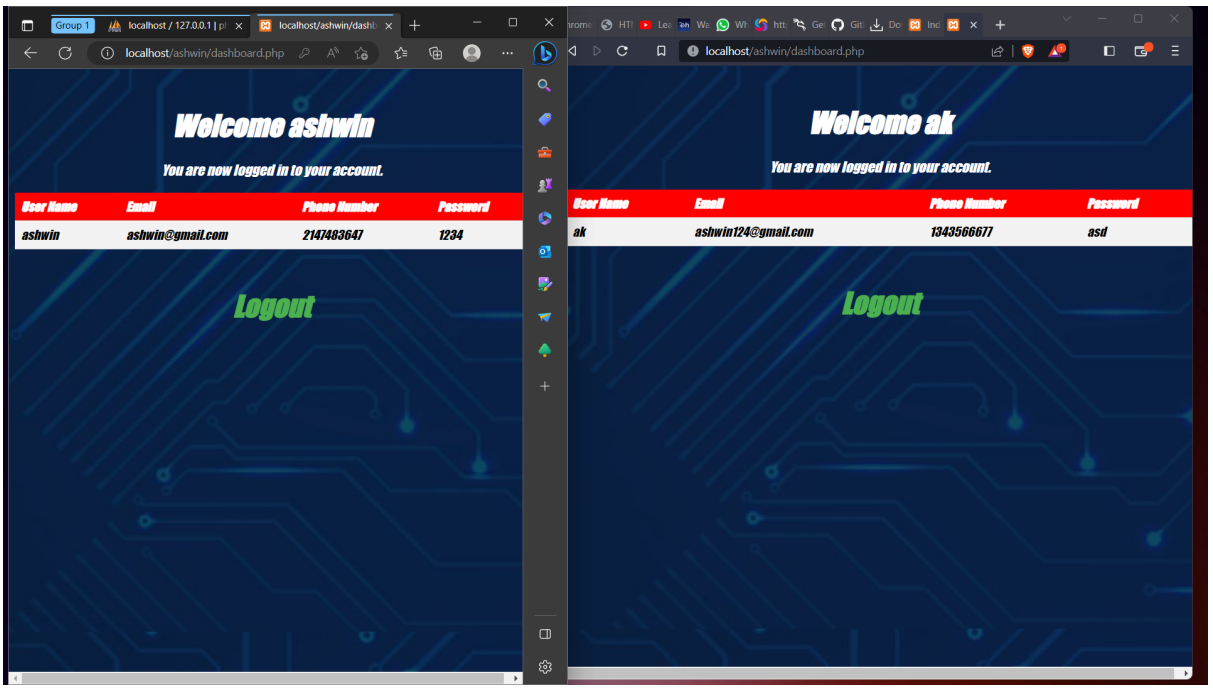
SESSION HIJACKING VULNERABILITY

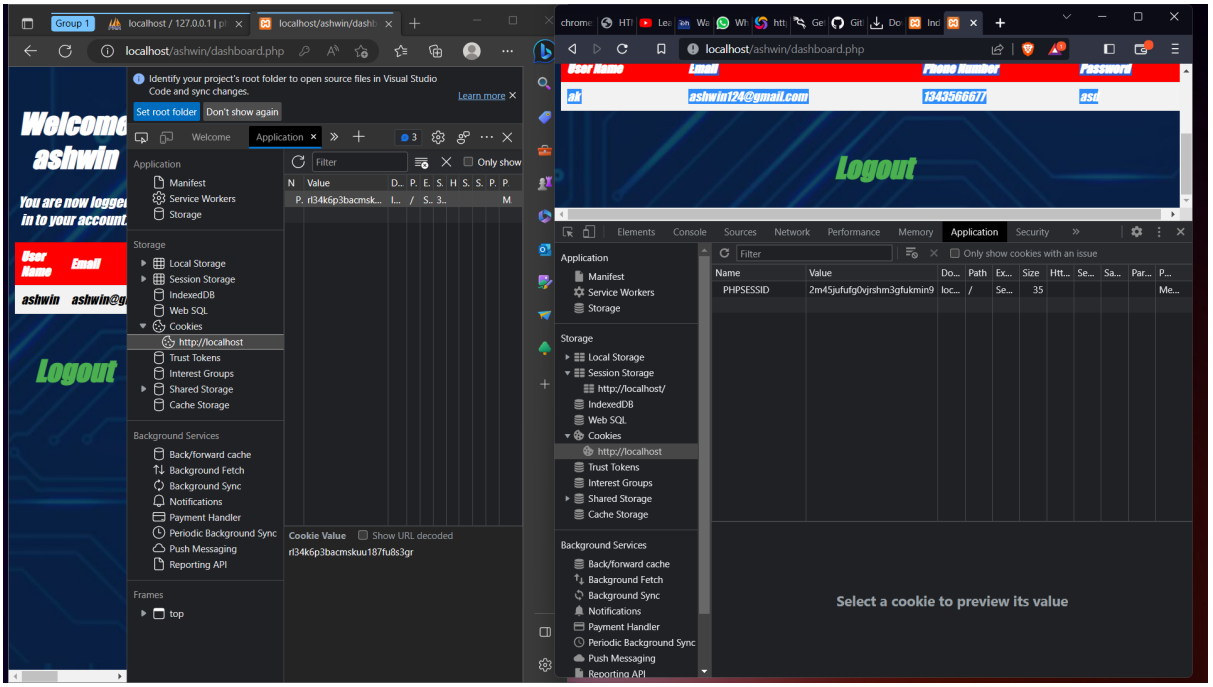
Session hijacking, also known as session fixation, is a type of security attack where a malicious user takes control of an authenticated session between a client and a server. The attacker can then impersonate the legitimate user and perform actions on their behalf, such as accessing sensitive information or performing unauthorized transactions.

Session hijacking typically occurs in the following way:

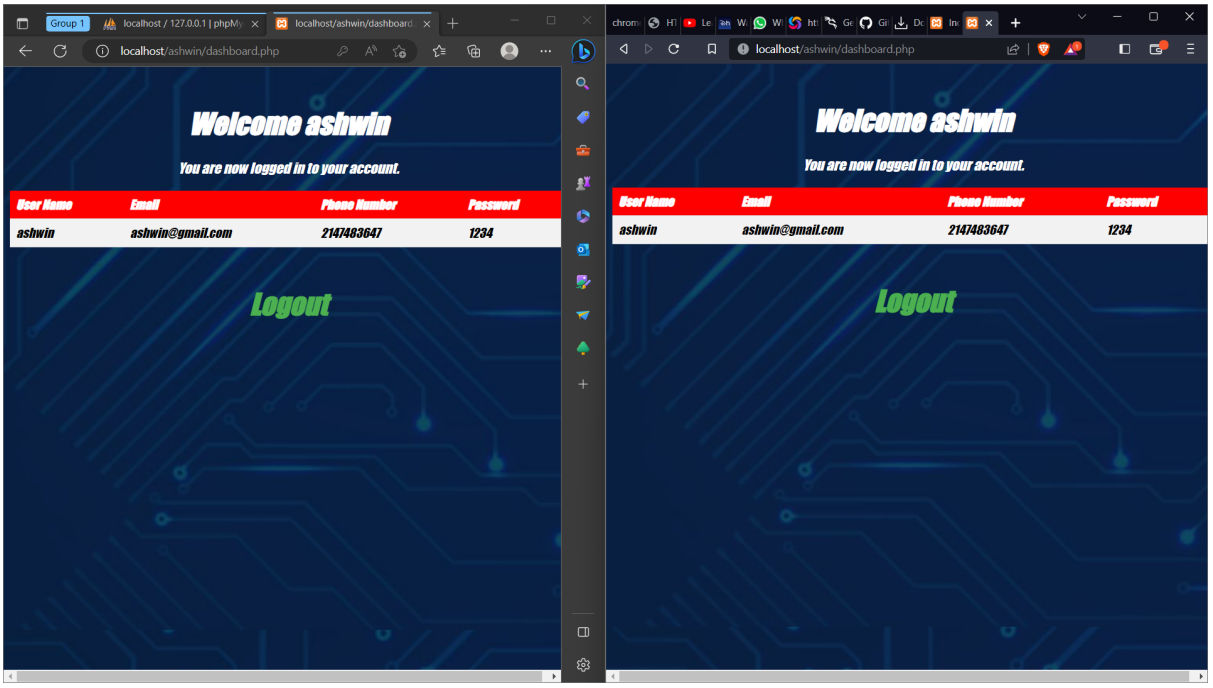
1. The legitimate user logs into a website or application.
2. The website or application generates a unique session ID for that user, which is stored in a cookie or in the URL.
3. The attacker intercepts the session ID, either by eavesdropping on the network or by stealing the cookie or URL.
4. The attacker uses the stolen session ID to impersonate the legitimate user and access their account.

There are several ways to prevent session hijacking, such as using HTTPS to encrypt the communication between the client and server, implementing secure session management techniques like expiring sessions after a certain period of inactivity or using randomly generated session IDs, and using multi-factor authentication to verify the user's identity.


Welcome ashwin
You are now logged in to your account.
User Name
Email
Phone Number
Password
ashwin
ashwin@gmail.com
2147483647
1234
Logout
Welcome ak
You are now logged in to your account.
User Name
Email
Phone Number
Password
ak
ashwin124@gmail.com
1343566677
asd
Logout


Welcome ashwin
You are now logged in to your account.
Logout
Identify your project's root folder to open source files in Visual Studio Code and sync changes.
Set root folder
Don't show again
Learn more
Application
Manifest
Service Workers
Storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
http://localhost
Trust Tokens
Interest Groups
Shared Storage
Cache Storage
Background Services
Back/forward cache
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Reporting API
Frames
top
Cookie Value
Show URL decoded
rl34k6p3bacmskuu187fu8s3gr
Select a cookie to preview its value

Output:



How to prevent session hijacking:

Session hijacking is a type of cyber attack where an attacker takes control of a user's session on a website or application, typically by stealing the user's session token or cookie. To prevent session hijacking, you can take the following steps:

1. Use HTTPS: Use HTTPS (HTTP Secure) to encrypt the traffic between the web server and the user's browser. This will make it more difficult for attackers to intercept and steal session cookies.
2. Implement secure session management: Use secure session management practices, such as generating random and unique session IDs, setting a timeout for sessions, and encrypting session data.

3. Use secure cookies: Use secure cookies that are encrypted and marked as HTTP-only. This will prevent them from being accessed by client-side scripts and reduce the risk of cross-site scripting attacks.
4. Implement multi-factor authentication: Implement multi-factor authentication, which requires users to provide multiple forms of identification before they can access their account. This