**Mini project**

**Title:** Design the Access control vulnerability

**Introduction:**

Unprotected admin functionality is a significant security vulnerability that can lead to unauthorized access, modification, and theft of sensitive data. It occurs when administrative functions of a software application are not properly secured, allowing unauthorized users to access and modify sensitive data.

This vulnerability can be caused by various factors, including the use of default or weak passwords for administrative accounts, lack of access controls, insufficient logging and monitoring, and lack of input validation for administrative functions.

To address this vulnerability, software developers should implement strong access controls, enforce strong password policies, log and monitor all administrative activities, and implement input validation for all user input. Regular testing and auditing of the application for vulnerabilities is also essential to ensure that it is secure.

Failure to address unprotected admin functionality can lead to significant financial and reputational damage, legal liability, and loss of trust from users and customers. Therefore, it is essential for software developers to prioritize security and take proactive measures to ensure that their applications are secure and protected from this vulnerability.

Some examples of unprotected admin functionality include:

* Default or weak passwords: Applications that use default or weak passwords for administrator accounts are vulnerable to unauthorized access.
* Lack of access controls: Applications that do not implement proper access controls for administrative functions can allow unauthorized users to access and modify sensitive data.
* Insufficient logging and monitoring: Applications that do not log and monitor administrative activities can make it difficult to detect and respond to unauthorized access or modification of data.
* Lack of input validation: Applications that do not validate user input for administrative functions can be vulnerable to injection attacks, where malicious code is injected into the application and executed.

To mitigate the risk of unprotected admin functionality, software developers should implement strong access controls, enforce strong password policies, log and monitor all administrative activities, and implement input validation for all user input. It is also important to regularly test and audit the application for vulnerabilities and implement patches and updates as needed to ensure that the application is secure.

**Portswigger:**

PortSwigger is a company that provides web application security solutions. It is best known for its flagship product, Burp Suite, which is an integrated platform for performing security testing of web applications. Burp Suite has become a standard tool in the industry and is used by security professionals, web developers, and researchers around the world.

In addition to Burp Suite, PortSwigger also provides a range of other web security tools and resources, including training courses, a web security academy, and a blog that covers the latest trends and techniques in web application security. The company is based in the UK and was founded by Dafydd Stuttard in 2004.

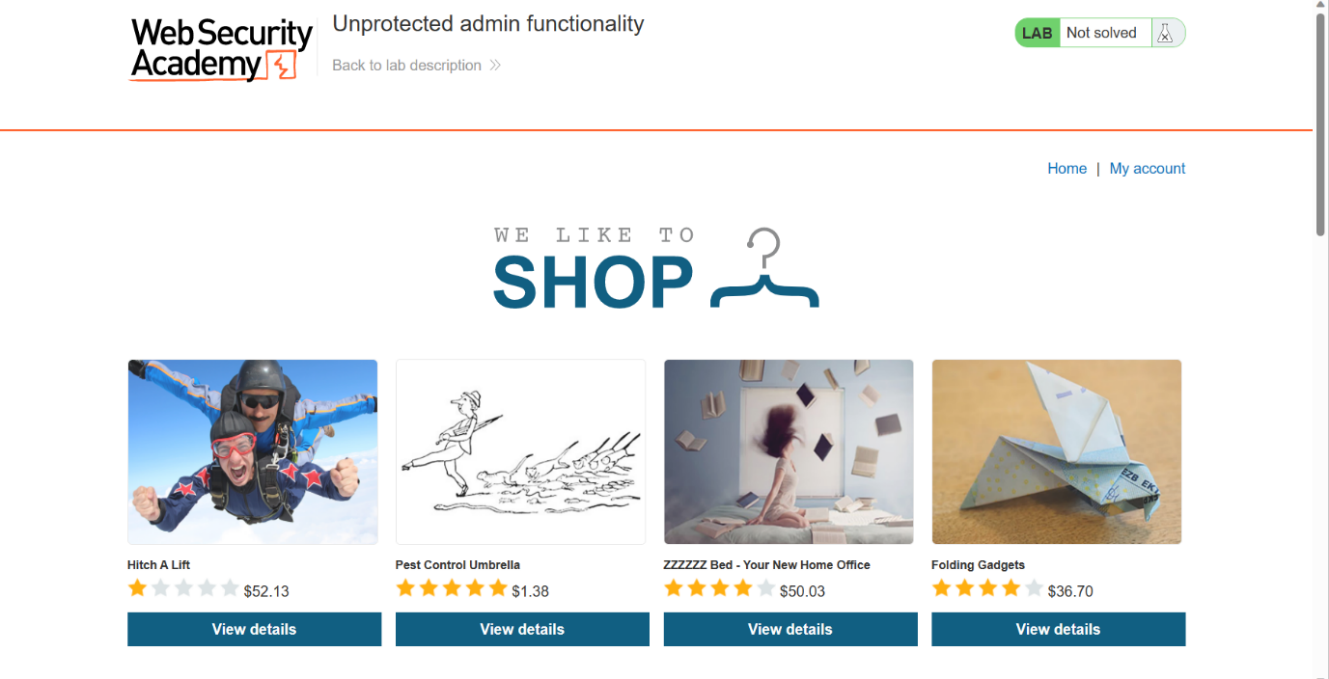
**Implementation:**

* Start by launching PortSwigger and opening the web application you want to test.
* In PortSwigger, go to the Proxy tab and turn on intercept mode. This will allow you to intercept the requests sent between the web application and the server.
* Access the login page of the web application and enter an incorrect username and password. Intercept the request sent to the server by clicking on the "Intercept" button in the Proxy tab.
* Right-click on the intercepted request and select "Send to Repeater". This will open the request in the Repeater tool.
* In the Repeater tool, modify the request to access the administrative panel without proper authentication. This can be done by changing the URL to the administrative panel, or by modifying the parameters in the request to bypass authentication.
* Send the modified request and check if you are able to access the administrative panel without proper authentication.
* If you are able to access the administrative panel without proper authentication, then the web application has unprotected admin functionality. This is a serious security vulnerability that can allow attackers to access and modify sensitive data.
* Test for other vulnerabilities such as weak password policies, lack of access controls, and insufficient logging and monitoring.
* Once vulnerabilities have been identified, report them to the application owner or developer and provide recommendations for remediation.

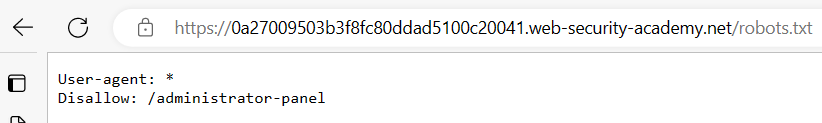
Testing for unprotected admin functionality using PortSwigger can help to identify potential security risks in web applications and encourage developers to implement proper security measures to protect against unauthorized access and modification of sensitive data.

**Testing and Result:**

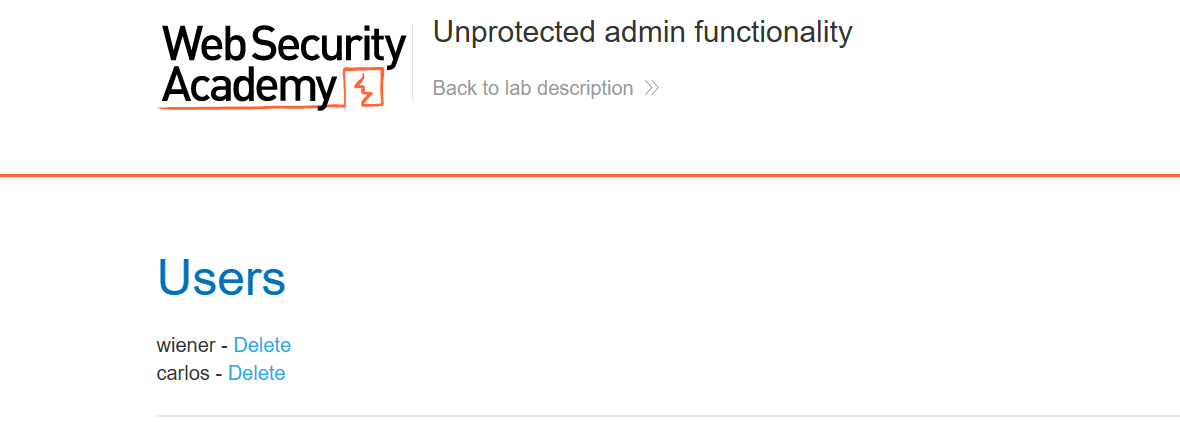
**Step 1:**

****

**Step 2:**

****

**Step 3:**

****

**Step 4:**

****

**Prevention:**

Some steps to prevent unprotected admin functionality are:

* Use strong password policies: Implement a strong password policy for all administrator accounts, requiring complex passwords and regular password updates.
* Implement access controls: Enforce access controls to restrict access to administrative functions to authorized users only.
* Validate user input: Implement input validation for all user input to prevent injection attacks, where malicious code is injected into the application and executed.
* Use encryption: Use encryption to protect sensitive data stored in the application and transmitted over the network.
* Log and monitor all activities: Log and monitor all administrative activities to detect and respond to unauthorized access or modification of data.
* Regularly test and audit the application: Regularly test and audit the application for vulnerabilities using PortSwigger or other security testing tools, and implement patches and updates as needed.

By implementing these steps, developers can prevent unprotected admin functionality and ensure that their applications are secure against potential security risks.

**Conclusion:**

In conclusion, leaving administrative functionality unprotected in a web application can pose a serious security risk. Unprotected admin functionality can allow an attacker to gain unauthorized access to sensitive information or perform malicious actions, such as modifying or deleting data, taking control of the application, or even compromising the underlying system.

Therefore, it is important to properly secure administrative functionality in web applications by implementing authentication and authorization mechanisms, enforcing strong passwords and access control policies, and regularly monitoring and auditing user activity.