Penetration Testing Report

Full Name :
Program : HCPT
Date :

Introduction

This report document hereby describes the proceedings and results of a Black Box security assessment conducted against the **Week {1} Labs**. The report hereby lists the findings and corresponding best practice mitigation actions and recommendations.

I. Objective

The objective of the assessment was to uncover vulnerabilities in the **Week {1} Labs** and provide a final security assessment report comprising vulnerabilities, remediation strategy and recommendation guidelines to help mitigate the identified vulnerabilities and risks during the activity.

II. Scope

The scope of the penetration testing project by Hacktify Cyber Security for clickjacking and HTML injection includes identifying vulnerabilities in the web application's frontend. Testing will focus on detecting clickjacking vulnerabilities that could lead to unauthorized actions by users. HTML injection points will be assessed to identify potential avenues for malicious code insertion. The boundaries of the project exclude testing of backend systems and network infrastructure. Results will be provided with recommendations for mitigation to enhance the application's security posture.

Application Name	{Lab 1 - Clickjacking} {Lab 2 – HTML Injection}

III. Summary

Outlined is a Black Box Application Security assessment for the **Week {1} and Week {2} Labs**.

Total number of Sub-labs: 8

High	Medium	Low
{1}	{3}	{4}

High	-	**1 Sub-lab with high difficulty level**
Medium	-	**3 Sub-labs with medium difficulty level**
Low	-	**4 Sub-labs with low difficulty level**

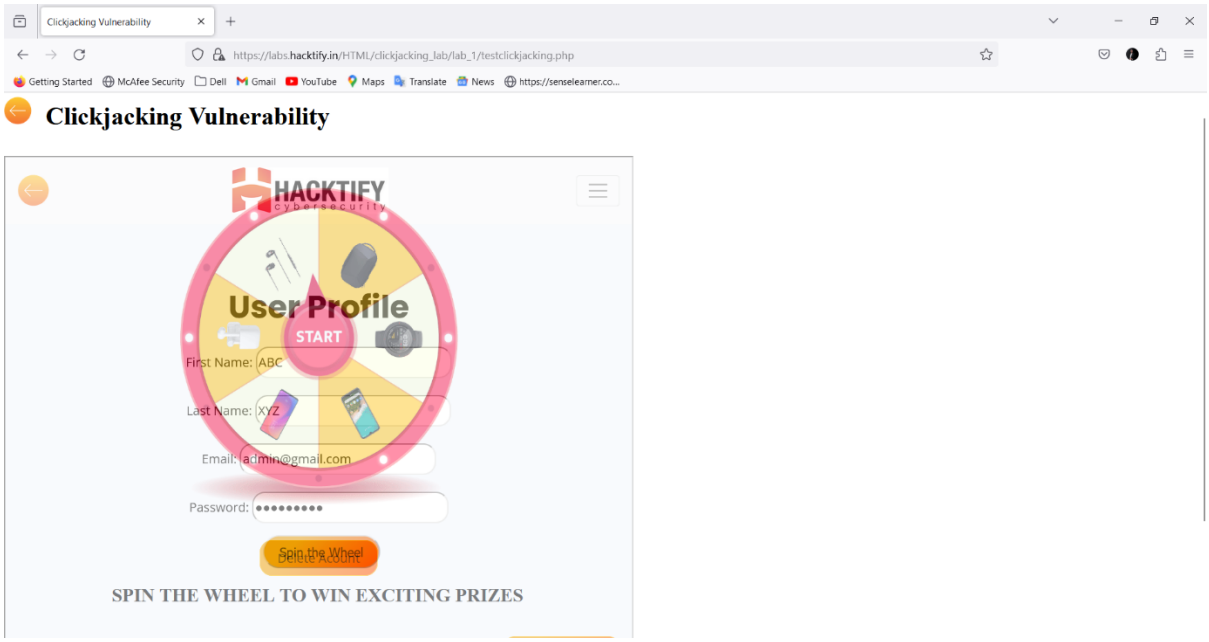
1. Clickjacking

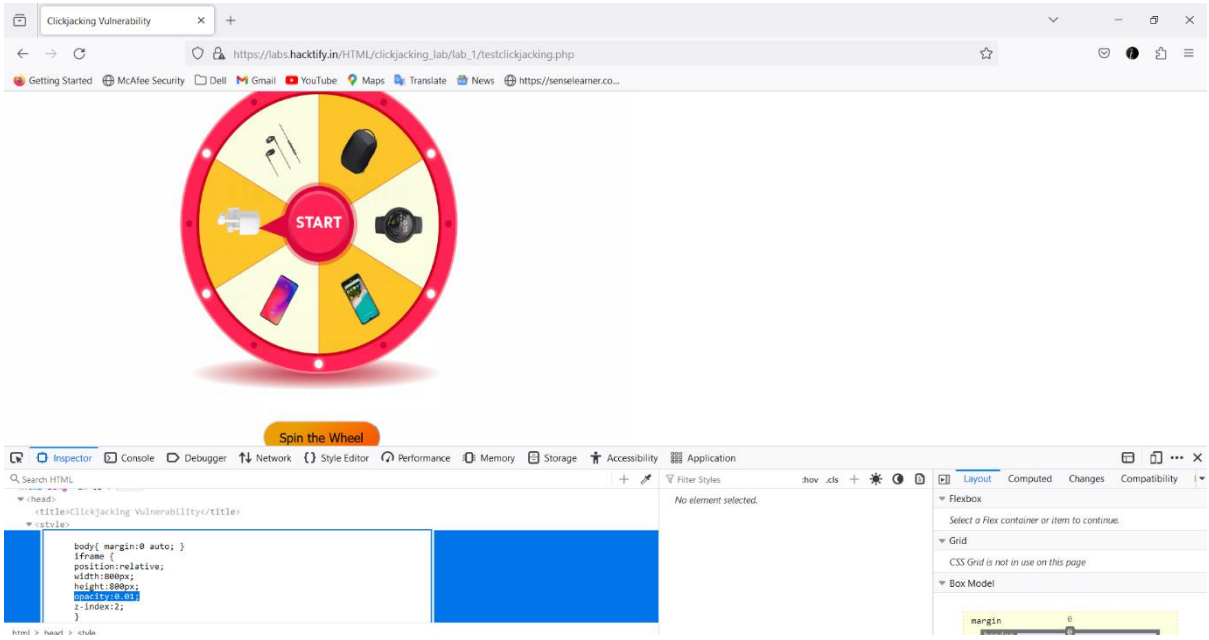
1.1. Let's Hijack!

Reference	Risk Rating
Sub-lab-1: Let's Hijack!	**Low**
Tools Used	
Browser "*Inspector*" is used to find the vulnerability.	
Vulnerability Description	
Clickjacking is a technique used by malicious actors to trick users into clicking on something different from what they perceive they are clicking on. This is typically done by overlaying transparent or opaque elements on top of legitimate buttons or links, so when a user clicks, they unknowingly interact with the hidden elements. While clickjacking can potentially lead to various security risks such as unauthorized actions performed by the user without their knowledge, it is often considered a low-risk threat compared to more severe forms of cyber-attacks. This is because clickjacking usually requires user interaction and may not directly lead to data breaches or system compromises. However, it can still be used as part of a broader attack strategy or to deceive users into unintended actions.	
How It Was Discovered	
Automated Tools – Browser Inspector	
Vulnerable URLs	
https://labs.hacktify.in/HTML/clickjacking_lab/lab_1/lab_1.php	
Consequences of not Fixing the Issue	
If this vulnerability is not patched. The user profile will gets deleted in one click, if the user is already logged into the application.	
Suggested Countermeasures	
X-Frame-Options: X-Frame-Options was originally introduced as an unofficial response header in Internet Explorer 8 and it was rapidly adopted within other browsers. The header provides the website owner with control over the use of iframes or objects so that inclusion of a web page within a frame can be prohibited with the deny directive: *X-Frame-Options: deny* Alternatively, framing can be restricted to the same origin as the website using the same origin directive *X-Frame-Options: same origin* **Content Security Policy (CSP):** Content Security Policy (CSP) is a detection and prevention mechanism that provides mitigation against attacks such as XSS and clickjacking. CSP is usually implemented in the web server as a return header of the form: *Content-Security-Policy: policy*	
References	
https://portswigger.net/web-security/clickjacking https://owasp.org/www-community/attacks/Clickjacking https://www.imperva.com/learn/application-security/clickjacking/	

Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab



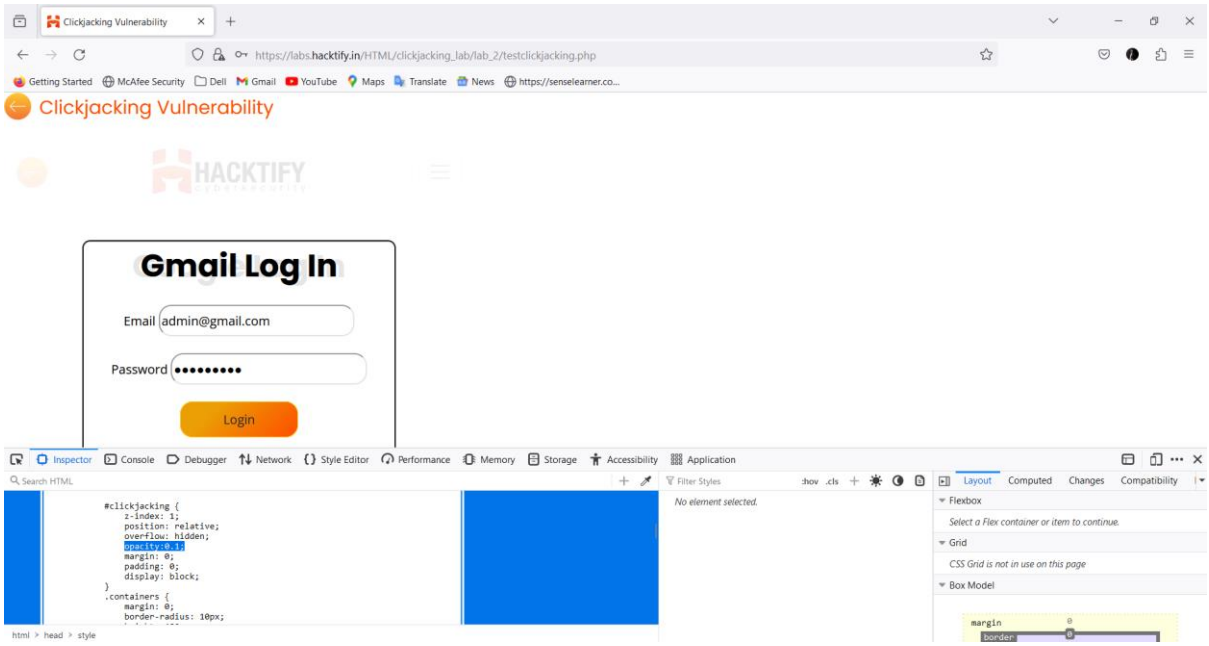


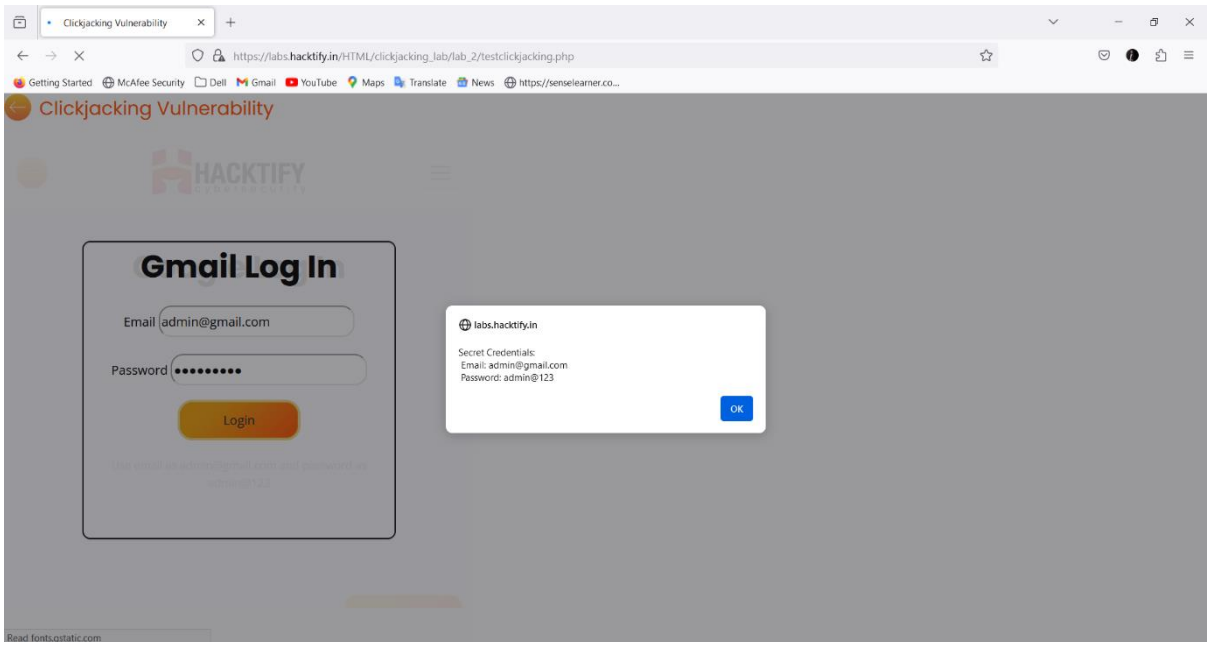
1.2. Re-Hijack!

Reference	Risk Rating
Sub-lab-2 Re-Hijack!	**Medium**
Tools Used	
Browser "*Inspector*" is used to find the vulnerability.	
Vulnerability Description	
Clickjacking is a technique used by malicious actors to trick users into clicking on something different from what they perceive they are clicking on. This is typically done by overlaying transparent or opaque elements on top of legitimate buttons or links, so when a user clicks, they unknowingly interact with the hidden elements. While clickjacking can potentially lead to various security risks such as unauthorized actions performed by the user without their knowledge, it is often considered a low-risk threat compared to more severe forms of cyber-attacks. This is because clickjacking usually requires user interaction and may not directly lead to data breaches or system compromises. However, it can still be used as part of a broader attack strategy or to deceive users into unintended actions.	
How It Was Discovered	
Automated Tools – Browser Inspector	
Vulnerable URLs	
https://labs.hacktify.in/HTML/clickjacking_lab/lab_2/lab_2.php	
Consequences of not Fixing the Issue	
It is showing like google login, but actually it is gmail login, so if the victim user login with mail id & password, Attacker will get the victim users gmail & password. By using gmail & password attacker will access victims gmail.	
Suggested Countermeasures	
X-Frame-Options: X-Frame-Options was originally introduced as an unofficial response header in Internet Explorer 8 and it was rapidly adopted within other browsers. The header provides the website owner with control over the use of iframes or objects so that inclusion of a web page within a frame can be prohibited with the deny directive: *X-Frame-Options: deny* Alternatively, framing can be restricted to the same origin as the website using the same origin directive *X-Frame-Options: same origin* **Content Security Policy (CSP):** Content Security Policy (CSP) is a detection and prevention mechanism that provides mitigation against attacks such as XSS and clickjacking. CSP is usually implemented in the web server as a return header of the form: *Content-Security-Policy: policy*	
References	
https://portswigger.net/web-security/clickjacking https://owasp.org/www-community/attacks/Clickjacking https://www.imperva.com/learn/application-security/clickjacking/	

Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab





2. HTML Injection

2.1. HTML's are easy!

Reference	Risk Rating
Sub-lab-1: HTML's are easy!	**Low**
Tools Used	
Browser "*View Page Sources*" is used to find the vulnerability.	
Vulnerability Description	
HTML injection is a type of web security vulnerability that allows an attacker to inject malicious HTML code into a webpage. This can occur when user input is not properly validated or sanitized before being included in the webpage's output. Attackers can exploit HTML injection vulnerabilities to manipulate the appearance or functionality of the webpage, steal sensitive information such as login credentials or session cookies, or redirect users to malicious websites. This type of vulnerability can have serious consequences and requires proper input validation and output encoding to mitigate the risk.	
How It Was Discovered	
Automated Tools – Browser View Page Sources (Ctrl + U)	
Vulnerable URLs	
https://labs.hacktify.in/HTML/html_lab/lab_1/html_injection_1.php	
Consequences of not Fixing the Issue	
If the vulnerability is not patched, an attacker can be able to write his own code to get the cookie information of the victim user, this HTML Injection attack leads to XSS attack.	
Suggested Countermeasures	
1. Implement strict input validation and sanitize user-supplied data. 2. Use contextual output encoding to prevent script execution. 3. Deploy Content Security Policy (CSP) to restrict script sources. 4. Educate developers on secure coding practices. 5. Regularly audit and test for vulnerabilities.	
References	
https://owasp.org/www-community/Injection_Information https://portswigger.net/web-security/cross-site-scripting/html-injection https://en.wikipedia.org/wiki/HTML_injection	

Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab