Penetration Testing Report

Full Name: Oluwatobi Samuel Akinlaja
Program: HCPT
Date: 22/07/2024

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.

1. 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.

2. Scope

This section defines the scope and boundaries of the project.

Application Name	{Open Redirect}, {HTML Injection}

3. Summary

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

Total number of Sub-labs: {count} Sub-labs

High	Medium	Low
{4}	{3}	{6}

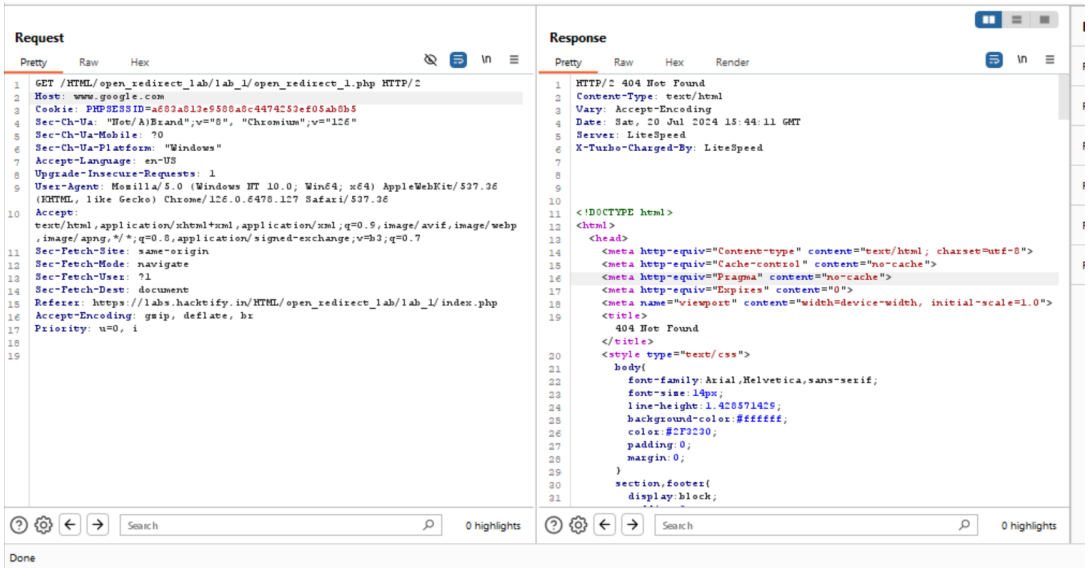
Sub lab with Unknown difficulty level - {1}

1. {OPEN REDIRECT}

1.1. {A SIMPLE HOST}

Reference	Risk Rating
A Simple Host	**Medium**
Tools Used	
Burp suite	
Vulnerability Description	
Sometimes, web applications use the referrer header to redirect users back to the page they were on before performing an action (like logging in or submitting a form). If the application blindly trusts the referrer header to determine where to redirect the user, it can be manipulated. An attacker can craft a URL with a malicious referrer header pointing to an external malicious site. This path suggests that the page `open_redirect_1.php` is vulnerable to an open redirect vulnerability.	
How It Was Discovered	
Automated Tools (Burp suite)	
Vulnerable URLs	
https://labs.hacktify.in/HTML/open_redirect_lab/lab_1/open_redirect_1.php	
Consequences of not Fixing the Issue	
It could lead to Phishing attacks, legal and compliance issues, loss of trust, reputational damage	
Suggested Countermeasures	
Set an appropriate referrer policy and content security policy, use of secure tokens, implementation of secure HTTP headers to control how browsers handles sensitive information and redirects	
References	
https://portswigger.net/web-security	

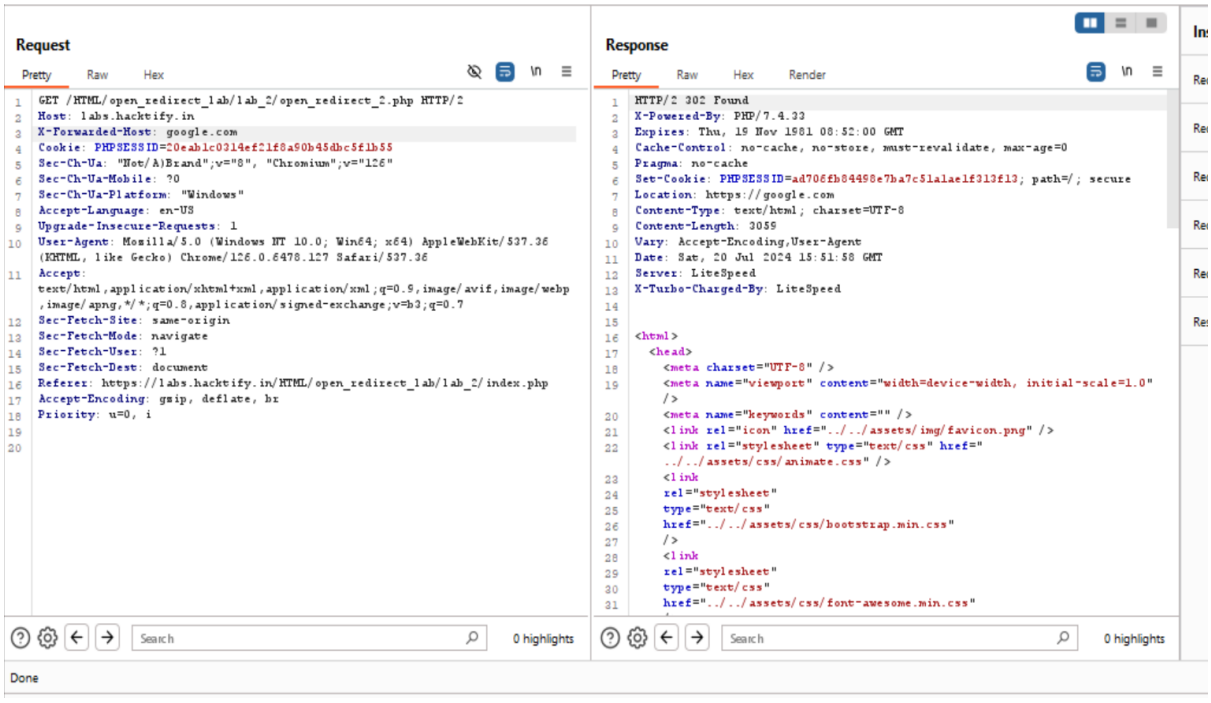
Proof of Concept



1.2. {STORY OF A BEAUTIFUL HEADER}

Reference	Risk Rating
Story of a Beautiful Header	Medium
Tools Used	
Burp Suite	
Vulnerability Description	
The referrer header is being used by the server to determine where to redirect the user after processing a request. As a result, the attacker can manipulate the referrer header to point to a malicious url. The path open_redirect_2.php indicates that the page may be susceptible to an open redirect vulnerability.	
How It Was Discovered	
Automated Tools (Burp Suite)	
Vulnerable URLs	
https://labs.hacktify.in/HTML/open_redirect_lab/lab_2/open_redirect_2.php	
Consequences of not Fixing the Issue	
This vulnerability could be exploited in phishing attacks, where an attacker creates a url that appears genuine but actually redirects users to a malicious website. Because the Referrer header usually displays a trusted domain (like labs.hacktify.in in this scenario), users might be less cautious about the redirection.	
Suggested Countermeasures	
Implement HTTP security headers, Use of safe redirection methods; This might entail implementing a whitelist of permitted redirection URLs or employing secure tokens to authenticate and authorize redirections.	
References	
https://portswigger.net/web-security	

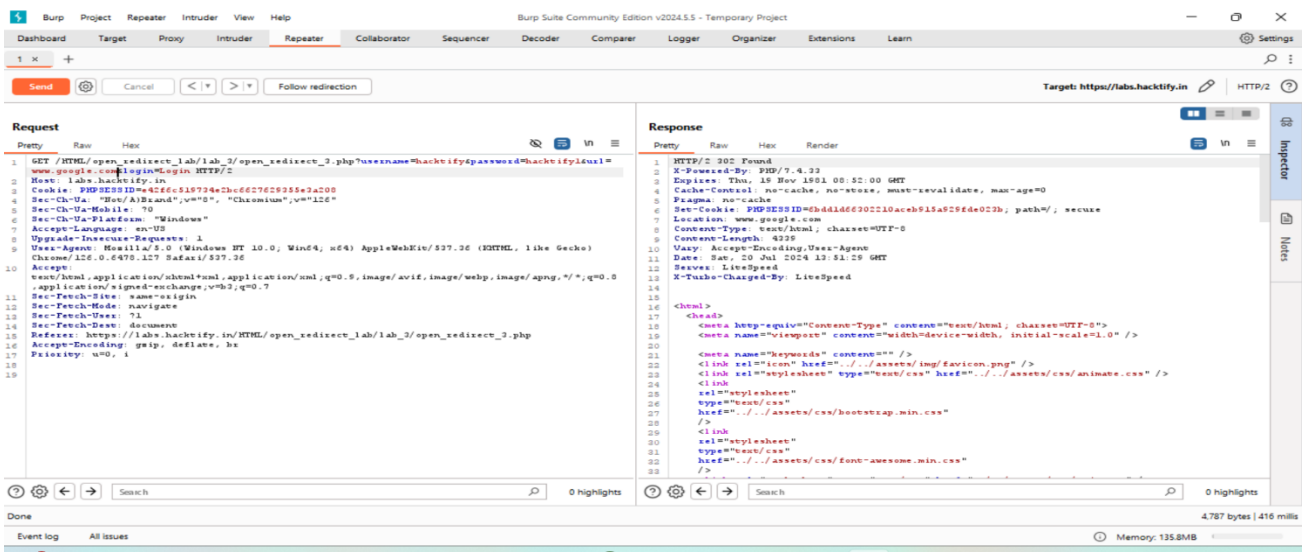
Proof of Concept



1.3. {SANITIZE PARAMS!!}

Reference	Risk Rating
Sanitize Params!!	**High**
Tools Used	
Burp Suite	
Vulnerability Description	
This URL includes parameters that may be susceptible to manipulation, particularly the parameter 'url=', which appears to define a URL (open_redirect_3_dashboard.php) that could be redirected to following a successful login. If attackers can modify this parameter to point to an external domain, it could result in an open redirect vulnerability.	
How It Was Discovered	
Automated Tools (Burp Suite)	
Vulnerable URLs	
url=open_redirect_3_dashboard.php, username, password, url, and login, PHPSESSID, Referrer, Content-Security-Policy,XFrameOptions,https://labs.hacktify.in/HTML/open_redirect_lab/lab_3/open_redirect_3.php?username=hacktify&password=hacktify1&url=open_redirect_3_dashboard.php&login=Login	
Consequences of not Fixing the Issue	
Failure to patch the vulnerabilities could result in unauthorized access to sensitive data, financial losses from fraud, legal liabilities, damage to reputation, and ongoing exploitation by malicious actors, posing significant operational and compliance risks to the organization.	
Suggested Countermeasures	
Developers and organizations can mitigate vulnerabilities by implementing secure coding practices such as input validation, robust session management, and encryption of sensitive data. They should also deploy security headers, conduct thorough testing for exploits, and perform regular security audits with updates to configurations to uphold web application integrity and security.	
References	
https://portswigger.net/web-security, https://owasp.org/, https://cve.mitre.org/,	

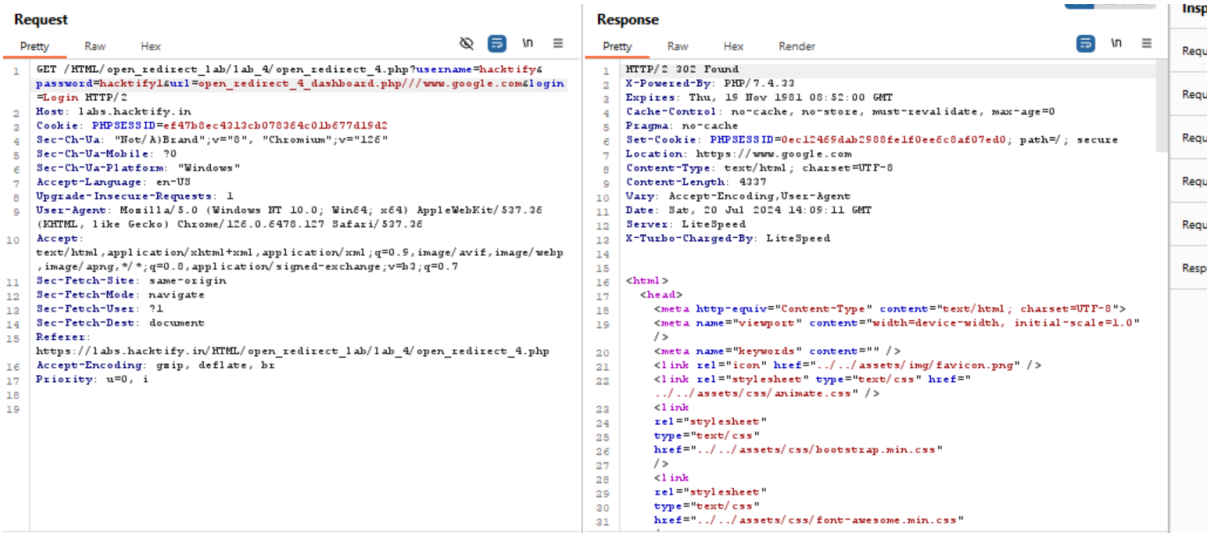
Proof of Concept



1.4. {PATTERNS ARE IMPORTANT}

Reference	Risk Rating
Patterns are Important	High
Tools Used	
Burp Suite	
Vulnerability Description	
The vulnerability originates from the url parameter (url=open_redirect_4_dashboard.php) included in the GET request. If the server script fails to validate or sanitize this parameter correctly, attackers can exploit it to redirect users to any external URLs of their choice. For instance, a manipulated URL could appear as follows: https://labs.hacktify.in/HTML/open_redirect_lab/lab_4/open_redirect_4.php?username=hacktify&password=hacktify1&url=https://malicious-site.com&login=Login	
How It Was Discovered	
Automated Tools (Burp Suite)	
Vulnerable URLs	
url=open_redirect_4_dashboard.php, https://labs.hacktify.in/HTML/open_redirect_lab/lab_4/open_redirect_4.php	
Consequences of not Fixing the Issue	
If the vulnerabilities are not patched, the consequences could include unauthorized access to sensitive data, potential financial losses from fraud, legal liabilities, damage to reputation, and ongoing exploitation by malicious actors.	
Suggested Countermeasures	
Implement rigorous input validation and sanitization, enforce secure session management protocols, encrypt sensitive data both during transmission and while at rest, and maintain regular updates and patches to software to effectively mitigate these vulnerabilities.	
References	
https://portswigger.net/web-security, https://cve.mitre.org/,	

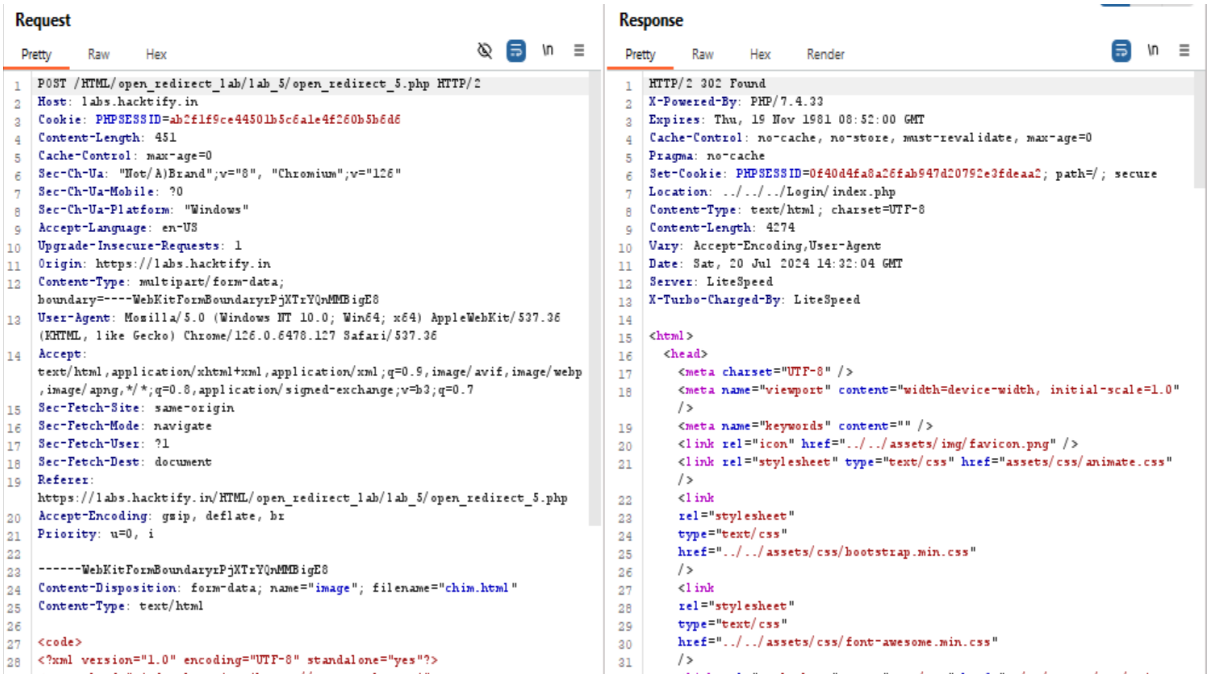
Proof of Concept



1.5. {FILE UPLOAD? REDIRECT IT!}

Reference	Risk Rating
{File Upload? Redirect it!}	High
Tools Used	
Burp Suite	
Vulnerability Description	
The vulnerability stems from the image form field within the multipart/form-data request, where SVG code such as <svg onload="window.location='https://www.google.com'"> can execute JavaScript upon image loading, potentially redirecting users to unintended URLs, posing an open redirect vulnerability.	
How It Was Discovered	
Automated Tools (Burp Suite)	
Vulnerable URLs	
https://labs.hacktify.in/HTML/open_redirect_lab/lab_5/open_redirect_5.php,<svgonload="window.location='https://www.google.com'">	
Consequences of not Fixing the Issue	
Failure to patch the vulnerability could lead to phishing attacks, malware distribution, reputation damage, regulatory issues, and ongoing exploitation by malicious actors.	
Suggested Countermeasures	
Implement strict input validation, sanitize user inputs rigorously, and enforce content security policies to mitigate open redirect vulnerabilities effectively.	
References	
https://portswigger.net/web-security, https://owasp.org/, https://cve.mitre.org/,	

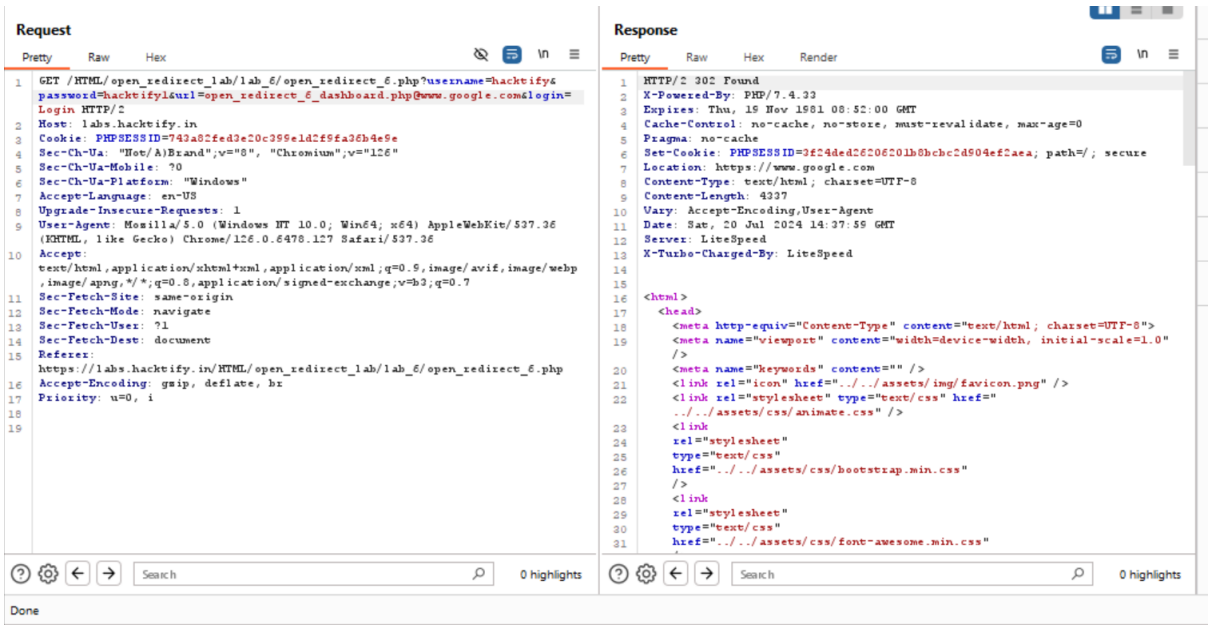
Proof of Concept



1.6. {SAME PARAM TWICE!}

Reference	Risk Rating
{Same Param Twice!}	**Medium**
Tools Used	
Burp Suite	
Vulnerability Description	
The vulnerability stems from the url parameter (url=open_redirect_6_dashboard.php) included in the GET request, which attackers can exploit to redirect users to any external URLs of their choice. For example, a manipulated URL could appear like this: https://labs.hacktify.in/HTML/open_redirect_lab/lab_6/open_redirect_6.php?username=hacktify&password=hacktify1&url=https://malicious-site.com&login=Login	
How It Was Discovered	
Automated Tools (Burp Suite)	
Vulnerable URLs	
https://labs.hacktify.in/HTML/open_redirect_lab/lab_6/open_redirect_6.php	
Consequences of not Fixing the Issue	
If the vulnerability in open_redirect_6.php is not patched, it could lead to attackers redirecting users to malicious websites, potentially compromising their sensitive information or facilitating further exploitation.	
Suggested Countermeasures	
To mitigate the open redirect vulnerability in `open_redirect_6.php`, ensure all user-supplied redirect URLs are validated against a whitelist of trusted domains before processing, and implement proper input sanitization and output encoding techniques.	
References	
https://portswigger.net/web-security, https://owasp.org/, https://cve.mitre.org/,	

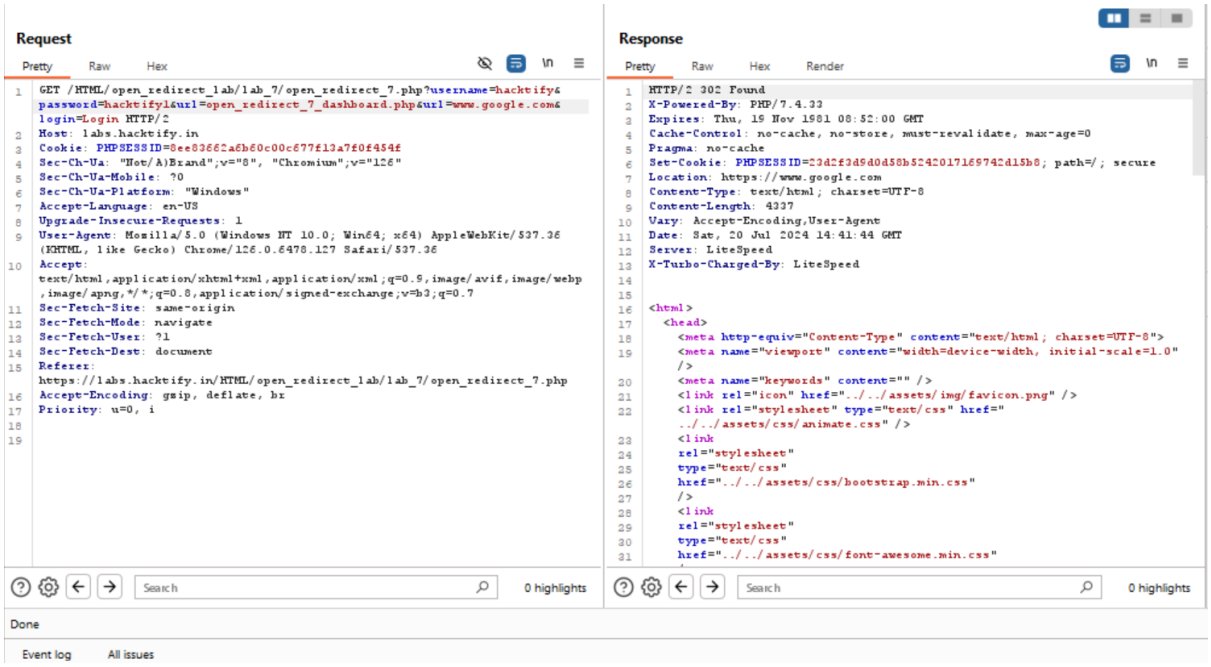
Proof of Concept



1.7. {DOMAINS? NOT ALWAYS!}

Reference	Risk Rating
{Domains? Not Always!}	**Medium**
Tools Used	
Burp Suite	
Vulnerability Description	
The vulnerability in GET /HTML/open_redirect_lab/lab_7/open_redirect_7.php lies in its handling of user-controlled input parameters (username, password, url) within the query string. These parameters are directly incorporated into the URL structure without proper validation or sanitization. This can allow an attacker to manipulate these parameters to redirect users to malicious websites (url parameter) or potentially exploit other vulnerabilities by injecting unexpected values into the system. This type of vulnerability, known as open redirect, can be used in phishing attacks or to gain unauthorized access to sensitive information.	
How It Was Discovered	
Automated Tools (Burp Suite)	
Vulnerable URLs	
https://labs.hacktify.in/HTML/open_redirect_lab/lab_7/open_redirect_7.php	
Consequences of not Fixing the Issue	
If the vulnerability is not patched, it could lead to attackers exploiting it to redirect users to malicious websites, potentially compromising their security or stealing sensitive information.	
Suggested Countermeasures	
To mitigate this open redirect vulnerability, ensure that all user-controllable input that determines the redirect destination is validated against a whitelist of allowed URLs or domains, and avoid relying solely on client-side mechanisms such as referer headers or JavaScript validation. Additionally, use server-side redirects with fixed, known URLs whenever possible to minimize the risk of unauthorized redirects.	
References	
https://portswigger.net/web-security, https://cve.mitre.org/,	

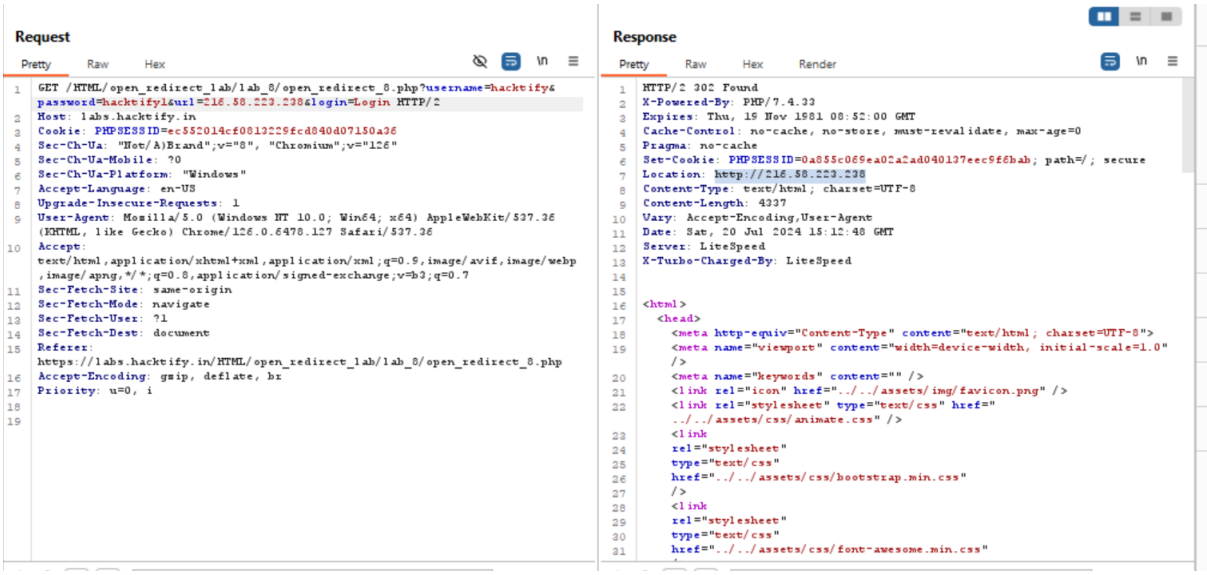
Proof of Concept



1.8 {STYLE DDIGIT SYMBOLS <3}

Reference	Risk Rating
{Style Digit Symbols <3}	**Low / Medium / High**
Tools Used	
Burp Suite, Kali Linux	
Vulnerability Description	
The vulnerability in open_redirect_8.php arises from improper validation or filtering of the url parameter in the query string (url=open_redirect_8_dashboard.php). This oversight allows an attacker to manipulate the url parameter to specify a redirect destination of their choice. By modifying the url parameter, an attacker can craft URLs that redirect users to malicious websites or other destinations outside the intended scope of the application. This vulnerability can be exploited for phishing attacks, spreading malware, or other malicious activities targeting unsuspecting users.	
How It Was Discovered	
Automated Tools (Burp Suite)	
Vulnerable URLs	
https://labs.hacktify.in/HTML/open_redirect_lab/lab_8/open_redirect_8.php	
Consequences of not Fixing the Issue	
If the vulnerability is not patched, it could lead to users being redirected to malicious websites, potentially compromising their security and privacy.	
Suggested Countermeasures	
To mitigate this vulnerability effectively, validate and sanitize all input parameters, particularly URLs, to enforce restrictions that allow only trusted domains, thereby preventing arbitrary redirects.	
References	
https://portswigger.net/web-security, https://cve.mitre.org/, https://owasp.org/	

Proof of Concept

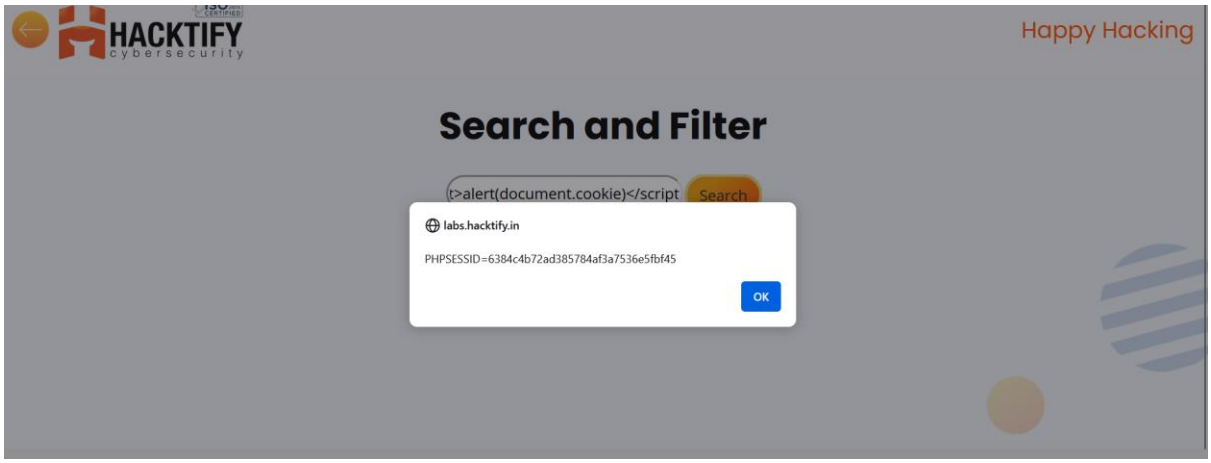


2. {HTML INJECTION}

2.1. {HTML'S ARE EASY}

Reference	Risk Rating
{Html's are easy}	Low
Tools Used	
Hacktify labs	
Vulnerability Description	
HTML, short for Hypertext Markup Language, serves as the standard markup language used to create web pages. A website is formed by a collection of these web pages. HTML elements are denoted by <> tags, with each tag serving a distinct function.	
How It Was Discovered	
Automated Tools	
Vulnerable URLs	
https://www.hacktifylab.com/learn/application-security/html-	
Consequences of not Fixing the Issue	
The link can be modified and the attacker access the user's credentials. It can also lead to security vulnerabilities such as cross-site scripting (XSS), enabling attackers to execute malicious scripts within a web application.	
Suggested Countermeasures	
Use proper encoding techniques to escape special characters (such as <, >, ", ', &) before outputting them to HTML.	
References	
https://cve.mitre.org/, https://hacktify.in/courses/hacktify-certified-pentester-hcpt/	

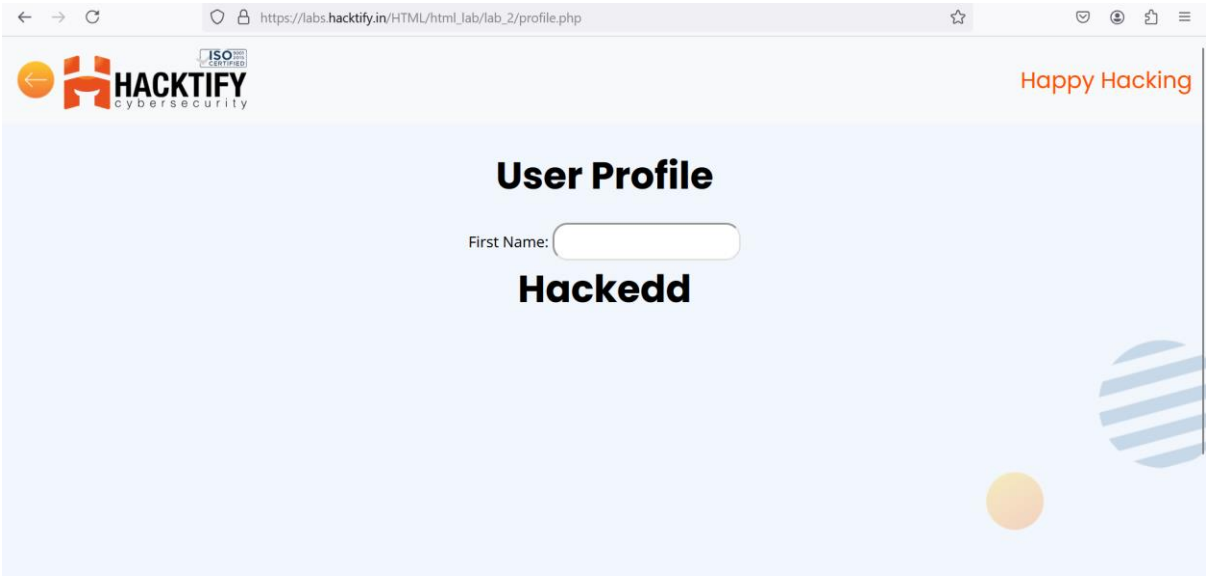
Proof of Concept



2.2. {LET ME STORE THEM}

Reference	Risk Rating
{Let Me Store Them}	**High**
Tools Used	
Hacktify labs	
Vulnerability Description	
The <form> element directs form data submission to profile.php via its action attribute. Users can input data into fields for first name, last name, email, pwd, and password. Notably, neither client-side nor server-side validation nor sanitization is in place for the <input> tags. An HTML comment <!--""/> immediately after the first name input implies a potential HTML injection attempt. This vulnerability could enable malicious users to inject arbitrary HTML or scripts into the first name field if left unaddressed	
How It Was Discovered	
Automated Tools	
Vulnerable URLs	
https://labs.hacktify.in/HTML/html_lab/lab_2/profile.php,<formmethod="POST"action="profile.php" >	
Consequences of not Fixing the Issue	
The impact of such vulnerabilities can range from compromising user data confidentiality (e.g., stealing passwords or personal information) to impacting the integrity and availability of the web application (e.g., modifying user profiles or deleting data).	
Suggested Countermeasures	
Implement strict server-side input validation in profile.php to enforce expected formats (e.g., alphanumeric names, valid email addresses), employ effective sanitization methods to remove or encode special characters from inputs, and use output encoding functions (e.g., htmlspecialchars() in PHP) to prevent HTML and script injection when displaying user data in HTML responses.	
References	
https://cve.mitre.org/, https://hacktify.in/courses/hacktify-certified-pentester-hcpt/	

Proof of Concept



2.3. {FILE NAMES ARE ALSO VULNERABLE}

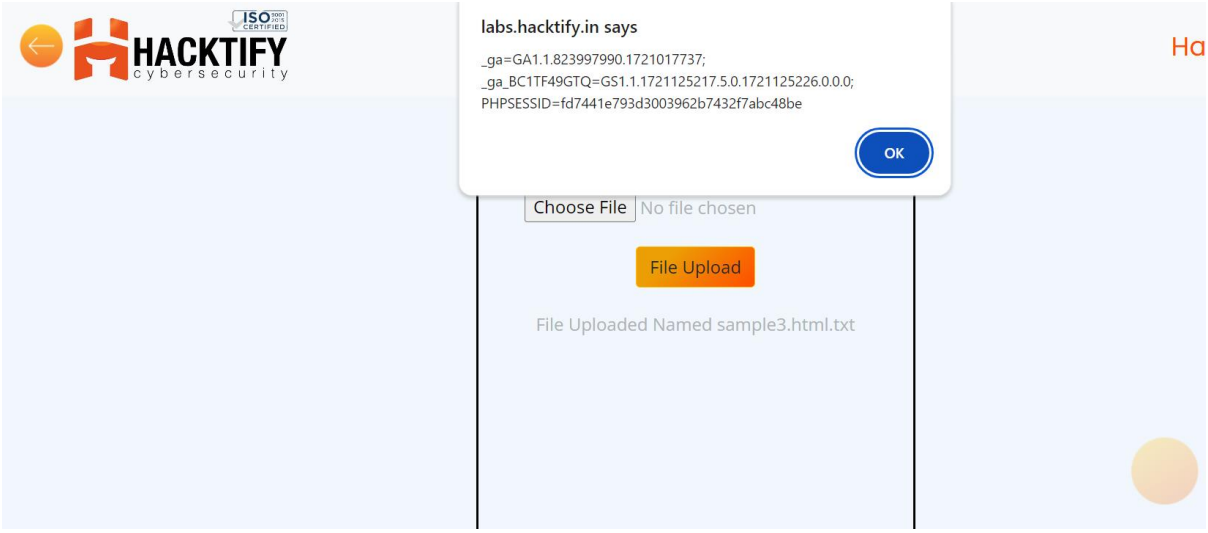
Reference	Risk Rating
{File Names are also Vulnerable}	**Medium**
Tools Used	
Hacktify labs and Burp Suite	
Vulnerability Description	
The <form> element is configured to submit data to html_injection_3.php when the form is submitted. The form uses enctype="multipart/form-data", indicating it is intended for file uploads. File uploads are sensitive because they allow users to submit files, which can potentially contain malicious content or be manipulated to exploit vulnerabilities on the server.	
How It Was Discovered	
Automated Tools	
Vulnerable URLs	
https://labs.hacktify.in/HTML/html_lab/lab_3/html_injection_3.php, POST/HTML/html_lab/lab_3/html_injection_3.php<formaction="html_injection_3.php"method="POST" enctype="multipart/form-data">	
Consequences of not Fixing the Issue	
If not properly validated and sanitized, an attacker could upload files with malicious content (e.g., scripts or executable files) that could be executed on the server. This can lead to various security issues such as remote code execution (RCE), denial of service (DoS), or unauthorized access to sensitive data.	
Suggested Countermeasures	
Implement comprehensive server-side measures for file uploads: validate file types and size, store files securely outside the web root directory, employ randomized or hashed filenames, set strict file permissions, enforce CSP headers, and sanitize file metadata to mitigate security risks effectively.	
References	
https://cve.mitre.org/, https://hacktify.in/courses/hacktify-certified-pentester-hcpt/	

Proof of Concept

2.4. {FILE CONTENT AND HTML INJECTION A PERFECT PAIR}

Reference	Risk Rating
{File Content and HTML Injection a Perfect Pair}	**Medium**
Tools Used	
Hacktify labs	
Vulnerability Description	
The <form> element is configured to submit data to html_injection_4.php when the form is submitted. The form uses enctype="multipart/form-data", indicating it is intended for file uploads. File uploads are sensitive because they allow users to submit files, which can potentially contain malicious content or be manipulated to exploit vulnerabilities on the server.	
How It Was Discovered	
Automated Tools	
Vulnerable URLs	
https://labs.hacktify.in/HTML/html_lab/lab_4/html_injection_4.php,<formaction="html_injection_4.php" method="POST" enctype="multipart/form-data">	
Consequences of not Fixing the Issue	
Insufficient validation and sanitization could allow attackers to upload files containing malicious content (such as scripts or executables) that may execute on the server, posing significant security risks such as remote code execution (RCE), potentially leading to server compromise or unauthorized access to stored data.	
Suggested Countermeasures	
Implement server-side validation for uploaded files to enforce type constraints (e.g., limited to image files) and size limits, verify file authenticity via extension checks and MIME type verification, store uploads outside the web root to prevent direct URL access, and apply strict permissions to mitigate unintended file execution or access.	
References	
https://cve.mitre.org/, https://hacktify.in/courses/hacktify-certified-pentester-hcpt/	

Proof of Concept



2.5. {INJECTING HTML USING URL}

Reference	Risk Rating
{Injecting HTML using URL}	**High**
Tools Used	
Hacktify labs	
Vulnerability Description	
This URL is vulnerable to HTML injection, as it incorporates a script tag <script>alert(document.cookie)</script> appended to the end of the URL path.	
How It Was Discovered	
Automated Tools	
Vulnerable URLs	
http://labs.hacktify.in/HTML/html_lab/lab_5/html_injection_5.php/<script>alert(document.cookie)</script>	
Consequences of not Fixing the Issue	
Accessing or rendering this URL in a browser triggers the execution of the embedded script within the webpage's context, posing a security risk such as cookie theft or other malicious activities. Manipulating the URL path to include script tags allows an attacker to potentially execute arbitrary JavaScript code on the client-side.	
Suggested Countermeasures	
Ensure that any user input or dynamic content used to construct URLs or HTML content undergoes thorough sanitization and validation to prevent the injection and execution of malicious scripts within the application.	
References	
https://cve.mitre.org/, https://hacktify.in/courses/hacktify-certified-pentester-hcpt/	

Proof of Concept

2.6. {ENCODE IT}

Reference	Risk Rating
{Encode IT}	**High**
Tools Used	
Hacktify labs and Burp Suite	
Vulnerability Description	
This indicates that the form data is being submitted to the server-side script html_injection_6.php located within the /HTML/html_lab/lab_6/ directory on the labs.hacktify.in domain. The vulnerability lies in how the search parameter is being handled, as it includes encoded HTML and JavaScript code (<script>alert(document.cookie)</script>), which can potentially lead to HTML injection and execution of arbitrary scripts on the server or client-side.	
How It Was Discovered	
Automated Tools	
Vulnerable URLs	
POST /HTML/html_lab/lab_6/html_injection_6.php, Your Searched results for <script>alert(document.cookie)</script>	
Consequences of not Fixing the Issue	
Failure to patch an HTML injection vulnerability could result in Cross-Site Scripting (XSS) attacks, enabling malicious actors to execute JavaScript in users' sessions, potentially leading to session hijacking, unauthorized actions, data theft, compromised user interactions, reputational damage, legal liabilities, and operational disruptions.	
Suggested Countermeasures	
Take immediate action to patch the vulnerability by implementing robust input validation, output encoding, and additional security measures to effectively mitigate risks of HTML injection and XSS attacks.	
References	
https://cve.mitre.org/, https://hacktify.in/courses/hacktify-certified-pentester-hcpt/	

Proof of Concept