

Penetration Testing Report

Full Name: Happy Jain

Program: HCPT

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	HTML Injection & Open Redirect
-------------------------	---

3. Summary

Outlined is a Black Box Application Security assessment for the **Week 1 Labs**.

Total number of Sub-labs: 14 Sub-labs

High	Medium	Low
4	4	6

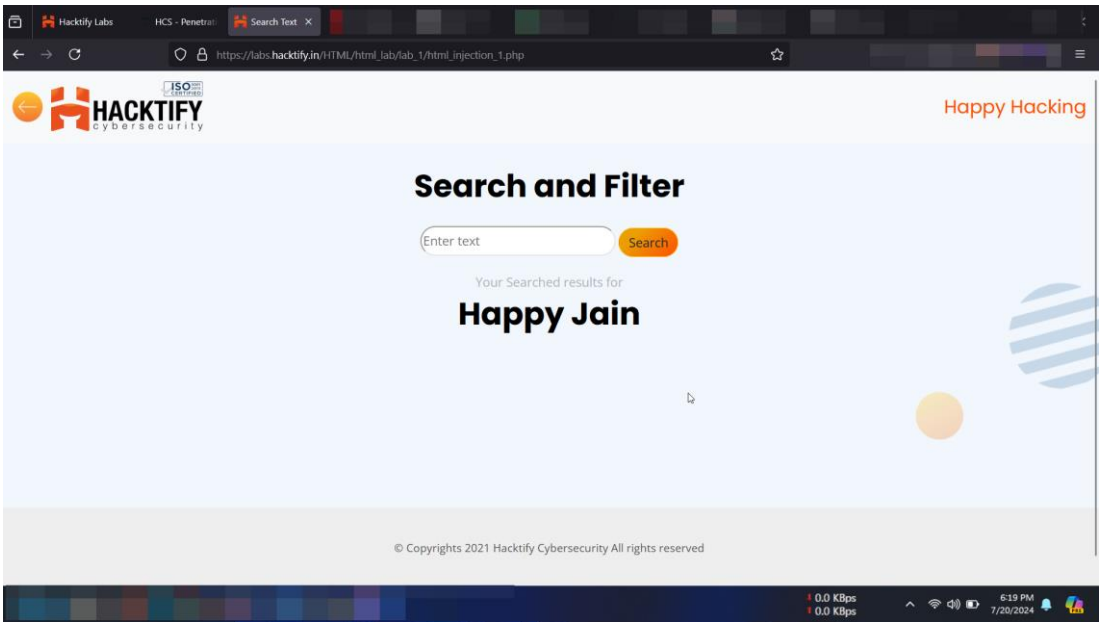
High - Number of Sub-labs with hard difficulty level

Medium - Number of Sub-labs with medium difficulty level

Low - Number of Sub-labs with Easy difficulty level

1. HTML Injection

1.1. HTML's Are Easy!

Reference	Risk Rating
HTML's Are Easy!	Low
Tools Used	
HTML Payload	
Vulnerability Description	
<ul style="list-style-type: none"> The addition of <code><h1>Happy Jain</h1></code> to a web page is an example of HTML injection. HTML injection occurs when untrusted user input is incorrectly included in a web page's output. In this case, the added code will be rendered as a level 1 heading ("Happy Jain") on the web page. If this input is not properly validated or sanitized, it can create a security vulnerability. 	
How It Was Discovered	
<ul style="list-style-type: none"> The addition of <code><h1>Happy Jain</h1></code> to a web page 	
Vulnerable URLs	
<ul style="list-style-type: none"> https://labs.hacktify.in/HTML/html_lab/lab_1/html_injection_1.php 	
Consequences of not Fixing the Issue	
<ul style="list-style-type: none"> Attackers can exploit this vulnerability to inject malicious code into a web page, steal sensitive information, or compromise the security of a web application. This can result 	

1.2. Let Me Store Them!

Reference	Risk Rating
Let Me Store Them!	Low
Tools Used	
HTML Payload	
Vulnerability Description	
<ul style="list-style-type: none"> Properly is the same as the first one. 	
How It Was Discovered	

- create a new user using the following values for each field,
First Name: ">< h1> Happy < /h1> "
Last Name: "">< h1>usr< /h1> "
Email-id: "happyusr@mail.com"
Password: "123"

- The HTML injection performed in lab 1 is called reflected HTML injection, as the payload entered in the search box persists only in the current browser session. In contrast, a stored HTML injection would persist even if the user logged out of their account. To test this vulnerability, you can create an account in one browser, such as Firefox, and then close the labs. Next, log in to the labs in another browser, such as Chrome, go to lab 2, and enter the same email ID and password ("happyusr@mail.com" and "123"). You can observe that the payload persists as stored HTML injection. Stored HTML injection allows the attacker to store the payload at the server-side and is more impactful than reflected HTML injection.

Vulnerable URLs

- https://labs.hacktify.in/HTML/html_lab/lab_2/register.php

Consequences of not Fixing the Issue

- Data breaches: Hackers can steal sensitive information, leading to data leaks and potential legal liabilities.
- Long-term infections: Exploiting unpatched vulnerabilities enables attackers to maintain persistent access to infected systems, potentially allowing them to launch further attacks

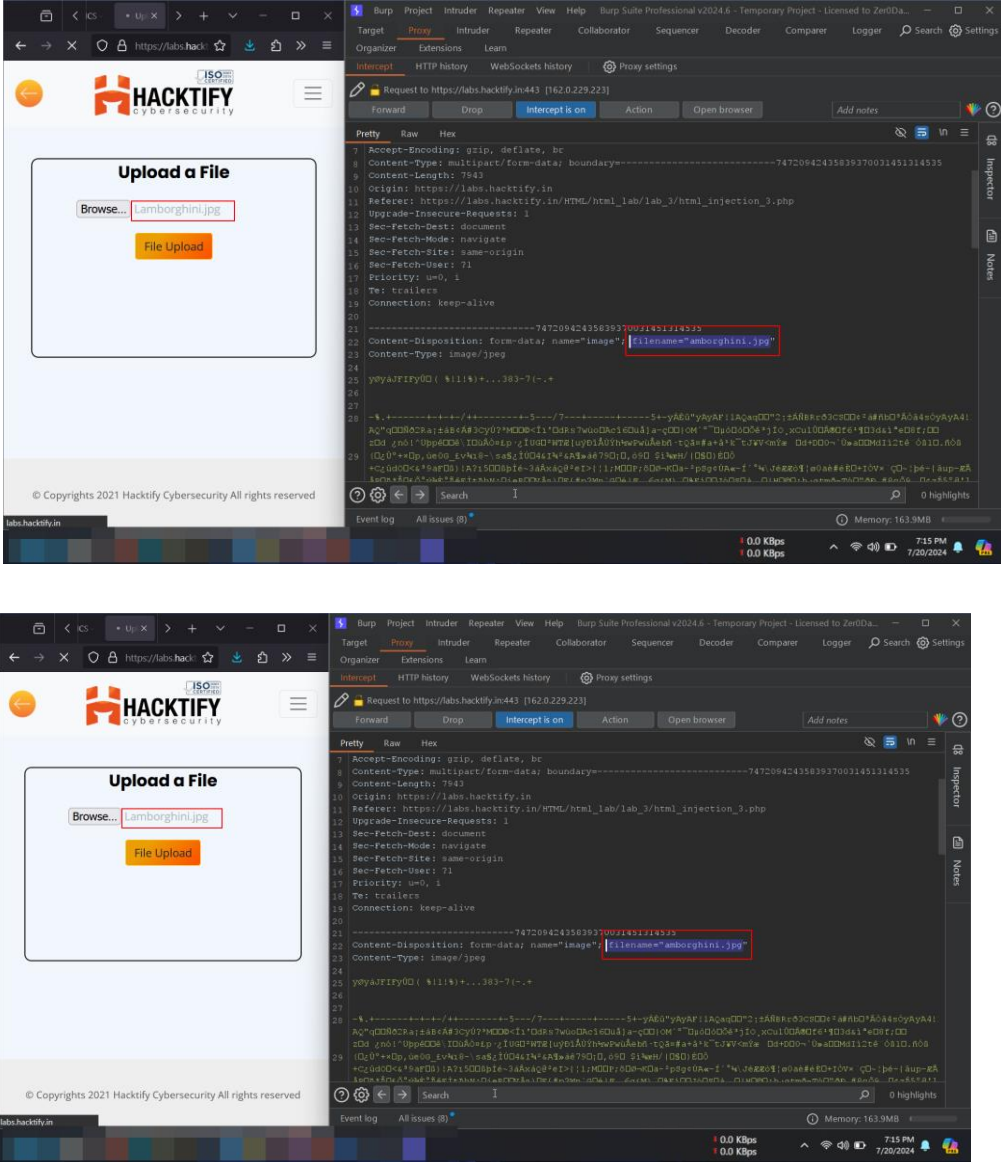
Suggested Countermeasures

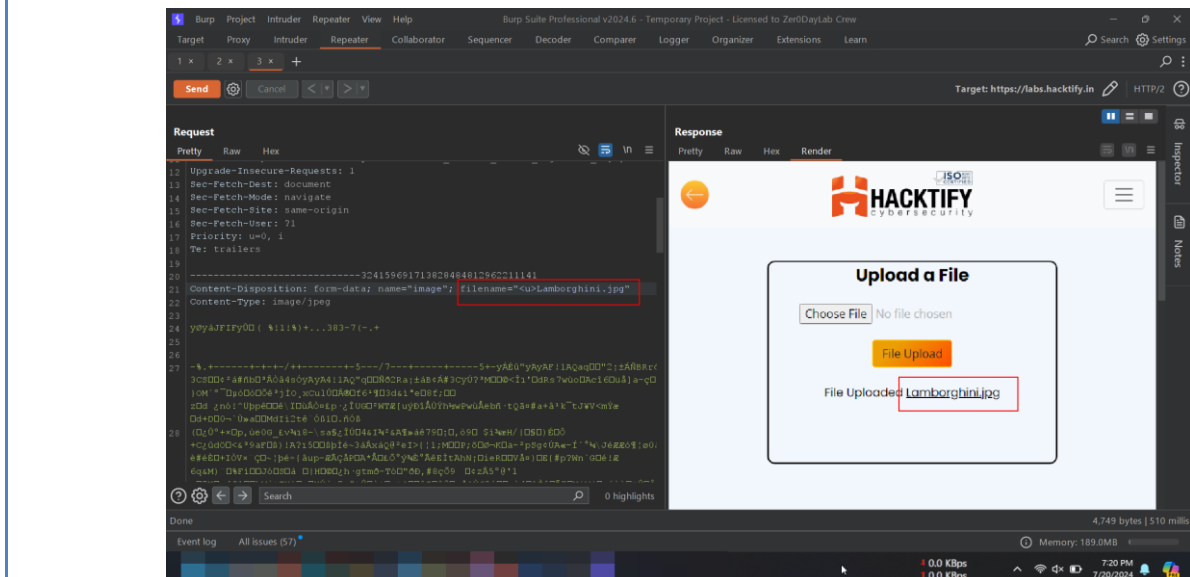
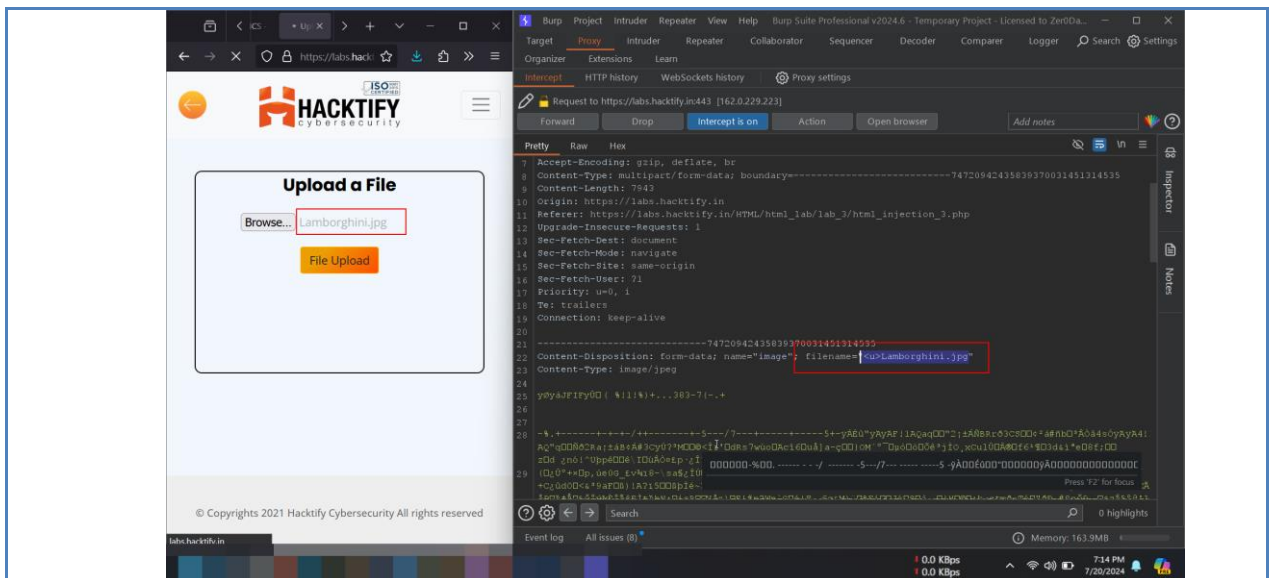
- Implement strict input validation: Validate user inputs to ensure that they meet the expected format and type. Use whitelisting approaches whenever possible to allow only trusted characters and values.
- Conduct regular security tests: Test your web applications for injection vulnerabilities during development, staging, and production phases. Automate security testing wherever possible to catch vulnerabilities early and often.
- Monitor and log activities: Log user interactions and events related to input validation and output encoding. Review logs periodically to detect anomalous behavior and investigate potential injection attempts.

References

- https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/03-Testing_for_HTML_Injection
- <https://www.acunetix.com/vulnerabilities/web/html-injection/>

1.3. File Names Are Also Vulnerable!

Reference	Risk Rating
File Names Are Also Vulnerable!	Low
Tools Used	
Burp Suite	
Vulnerability Description	
<ul style="list-style-type: none">• Insertion of client-side scripts into the code<ul style="list-style-type: none">- These vulnerabilities occur when untrusted user inputs are incorrectly included in a web page's output, enabling attackers to inject malicious JavaScript or other client-side scripts into the web application.	
How It Was Discovered	
 <p>The image displays two screenshots of the Burp Suite interface, illustrating a file upload request. The left screenshot shows the 'Raw' tab of the HTTP history, where the request body is visible. The right screenshot shows the 'Pretty' tab, which formats the request body for readability. Both screenshots show a file upload request to the URL 'https://labs.hacktify.in/443'. The request body contains the filename 'Lamborghini.jpg'.</p>	



- This observation suggests that the "filename" parameter, used to display the name of the uploaded file, lacks proper sanitization. Consequently, this allows the insertion of client-side scripts into the code.

Vulnerable URLs

- https://labs.hacktify.in/HTML/html_lab/lab_3/html_injection_3.php

Consequences of not Fixing the Issue

- Cross-Site Scripting (XSS) Attacks: Attackers can exploit the vulnerability to inject malicious scripts into web pages, potentially leading to the theft of sensitive user data, session hijacking, or defacement of the website.
- Cross-Site Scripting (XSS) Attacks: Attackers can exploit the vulnerability to inject malicious scripts into web pages, potentially leading to the theft of sensitive user data, session hijacking, or defacement of the website.

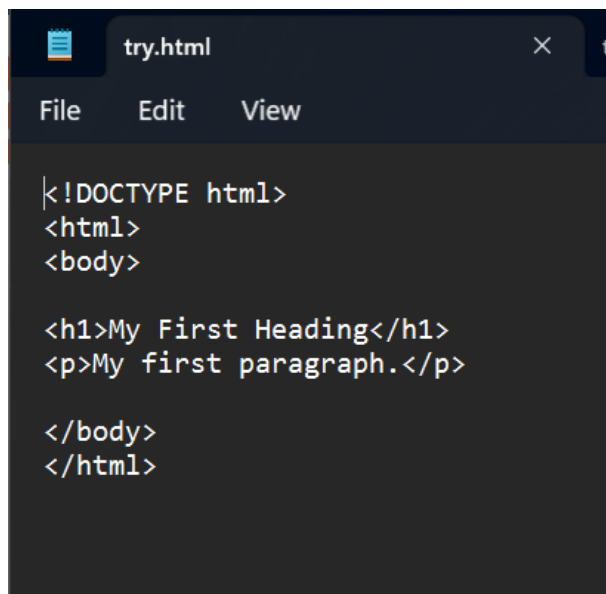
Suggested Countermeasures

- Input Validation: Implement strict input validation to filter out malicious script tags and other unwanted characters.
- Parameterized Queries: Use parameterized queries instead of direct string interpolation to prevent SQL injection and XSS attacks.
- Access Control: Restrict access to sensitive areas of the application to authorized personnel only.

References

1.4. File Content and HTML Injection A Perfect Pair!

Reference	Risk Rating
File Content and HTML Injection A Perfect Pair!	Medium
Tools Used	
Basic HTML code & Upload	
Vulnerability Description	
<ul style="list-style-type: none"> • The backend currently lacks a mechanism to verify the file type during the upload process, and additionally allows parsing and execution of files without proper validation. In analogous real-world scenarios, this vulnerability could potentially facilitate a shell upload, consequently granting unauthorized access to the server for the attacker. 	
How It Was Discovered	



```

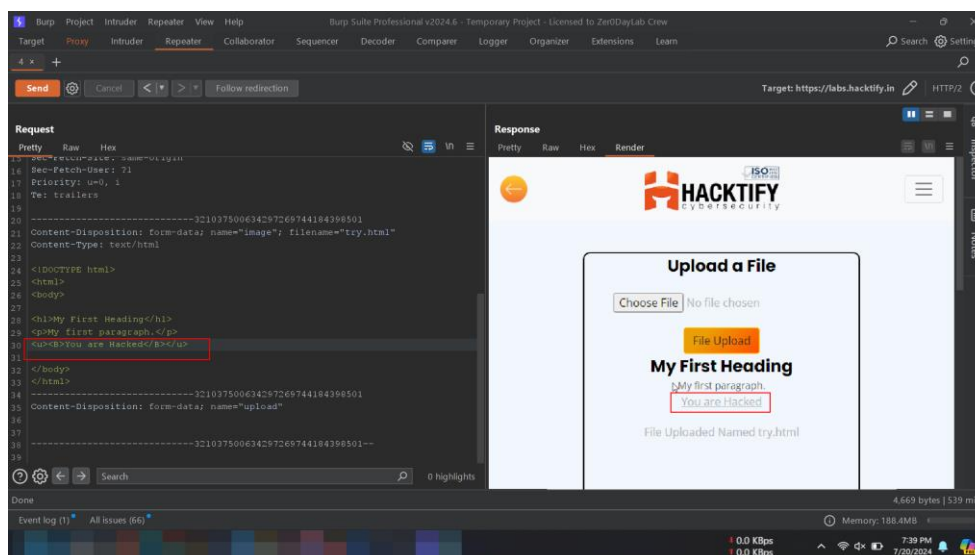
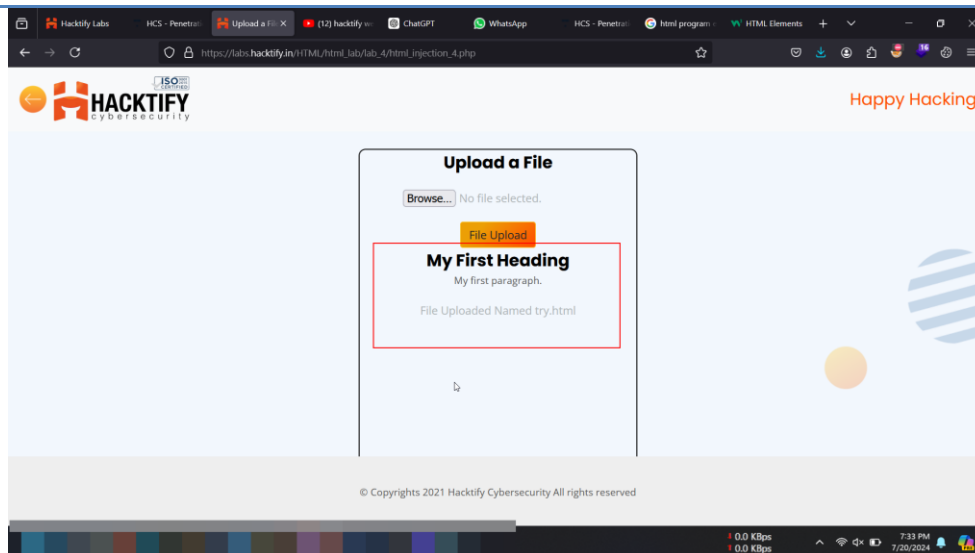
try.html
File Edit View

<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>

```



Vulnerable URLs

- https://labs.hacktify.in/HTML/html_lab/lab_4/html_injection_4.php

Consequences of not Fixing the Issue

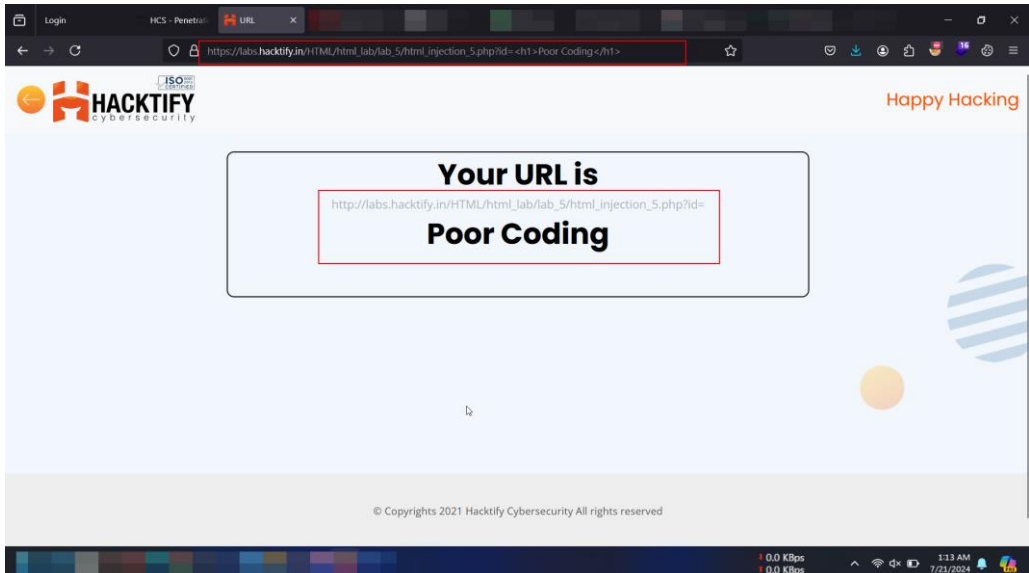
- **Unauthorized Access:** Exploiting the identified vulnerability may grant unauthorized individuals access to sensitive areas of the system or confidential data.
- **Data Compromise:** The absence of proper file type validation and execution control could result in the compromise of critical data, leading to data breaches or leakage.
- **Malicious Code Execution:** Allowing unverified files to be parsed and executed opens the door to the injection of malicious code, potentially leading to the manipulation or disruption of system functionalities.

Suggested Countermeasures

- Implement robust file type validation mechanisms during the upload process to ensure that only permitted file types are accepted.
- Utilize file signature checks or MIME type verification to enhance the accuracy of file type validation.
- Restrict the execution of uploaded files to only essential and safe functionalities. Avoid allowing arbitrary execution of uploaded files.

References

1.5. Injecting HTML Using URL

Reference	Risk Rating
Injecting HTML Using URL	Medium
Tools Used	
HTML Payload	
Vulnerability Description	
<ul style="list-style-type: none"> • The queries in the URL are not being filtered/sanitized which is the cause for our URL to be susceptible to HTML injection attacks. 	
How It Was Discovered	
<p>By adding payload: “?id=<h1>Poor Coding< /h1>”</p> 	

Vulnerable URLs

- https://labs.hacktify.in/HTML/html_lab/lab_5/html_injection_5.php

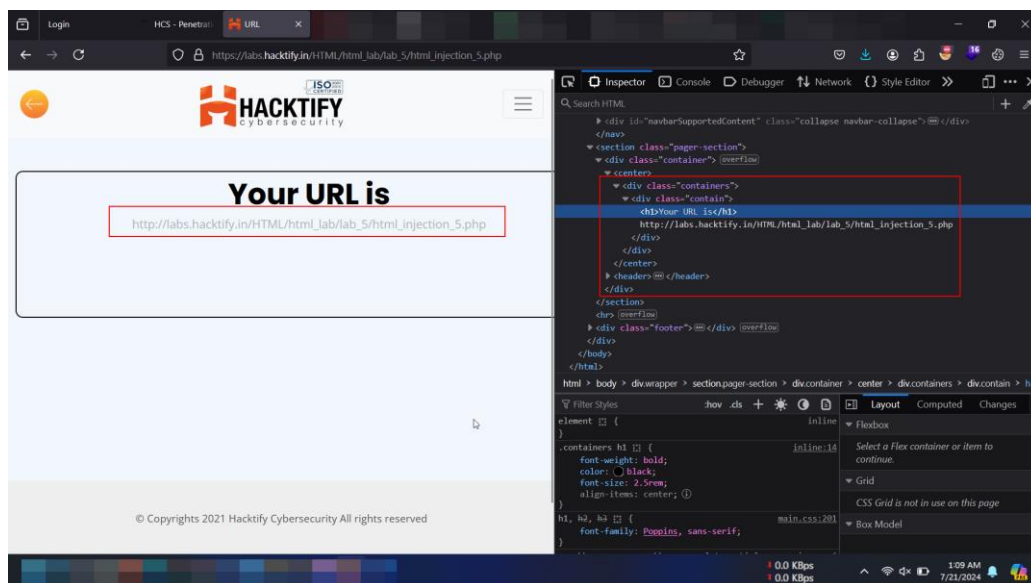
Consequences of not Fixing the Issue

- Cross-Site Scripting (XSS) Attacks: Attackers can exploit the vulnerability to inject malicious scripts into web pages, potentially leading to the theft of sensitive user data, session hijacking, or defacement of the website.
- Compromised User Trust: Security incidents resulting from XSS attacks can damage the trust that users have in the affected organization's website and services.
- Data Breaches: Unfixed XSS vulnerabilities can result in the compromise of sensitive information, such as user credentials, personal data, and financial details.

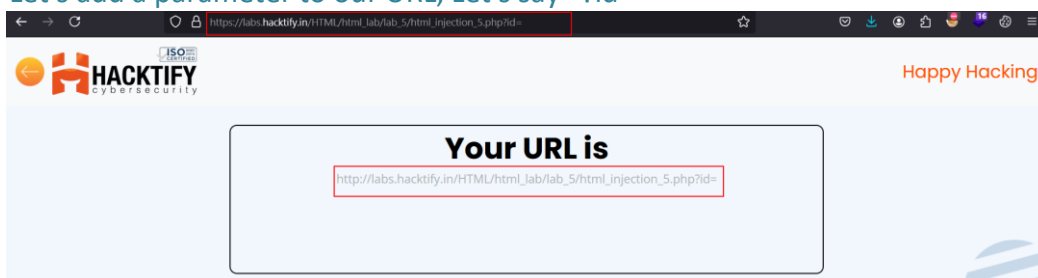
Suggested Countermeasures

- ## References

1- Check the source code for the lab.

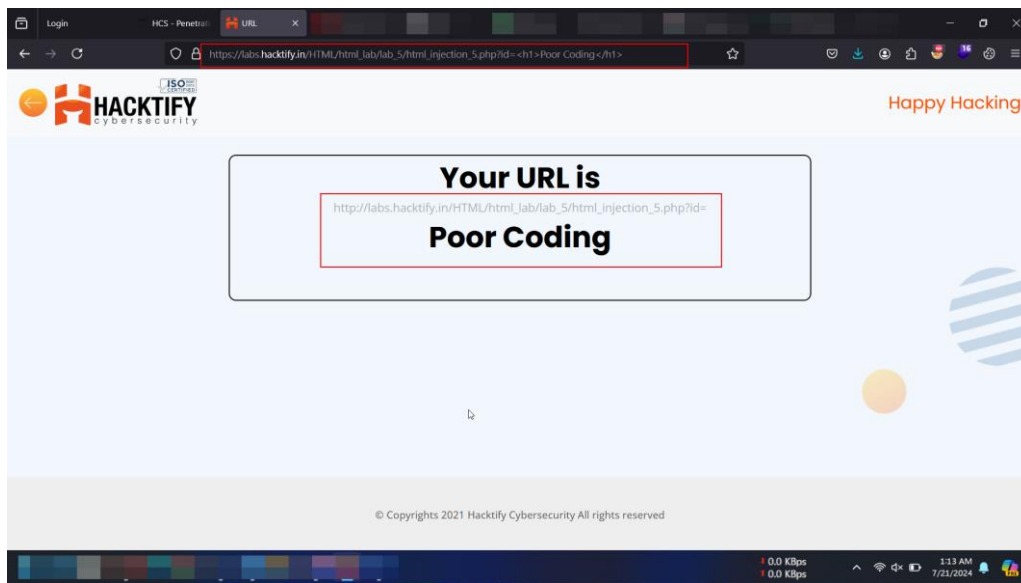


2- Let's add a parameter to our URL, Let's say "?id="

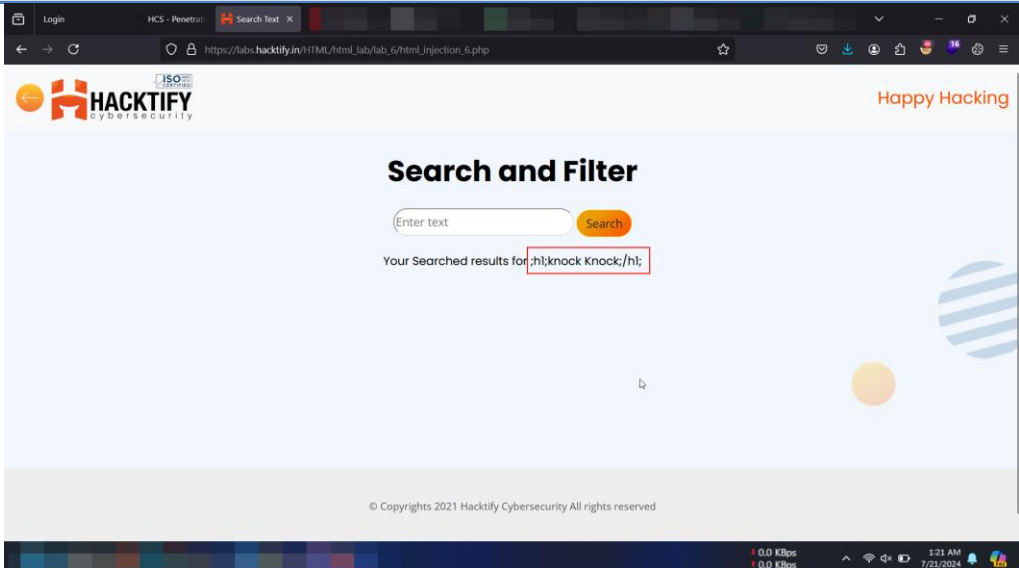
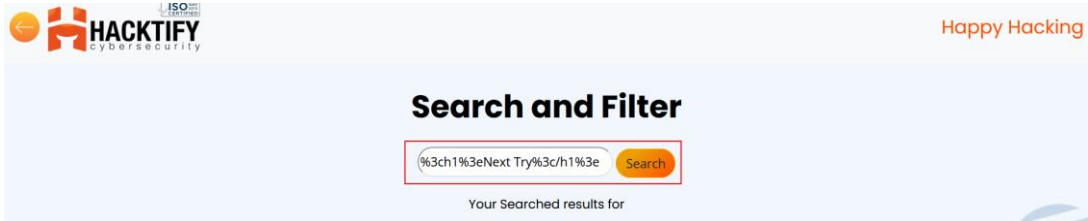


- That gives us the most important information that our UI is getting fetched from the server-side script to the URL we search for!!!

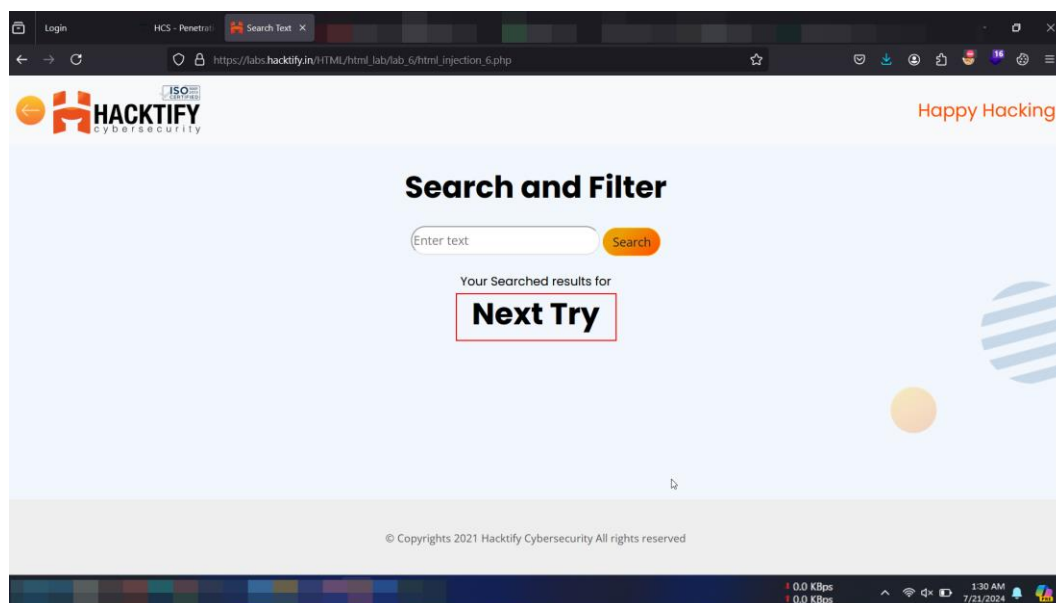
3- Now let's use the following payload: “?id=<h1>Poor Coding</h1>”



1.6. Encode IT!

Reference	Risk Rating
Encode IT!	Hard
Tools Used	
HTML Payload & URL encoding	
Vulnerability Description	
<ul style="list-style-type: none">• Blacklisting: creating a list of known malicious HTML tags, attributes, or other content and then blocking any input that matches items on the list. While blacklisting can be used as a temporary measure to mitigate HTML injection vulnerabilities, it is generally considered less effective than whitelisting.	
How It Was Discovered	
 <p>The screenshot shows a web browser window with the URL <code>https://labs.hacktify.in/html_lab/lab_6/html_injection_6.php</code>. The page has a header with the Hacktify logo and 'Happy Hacking' text. The main section is titled 'Search and Filter' and contains a search input field with the placeholder 'Enter text' and a 'Search' button. Below the input field, it says 'Your Searched results for' followed by the text <code>h!;knock Knock/h!</code> which is highlighted with a red box. The footer of the page mentions '© Copyrights 2021 Hacktify Cybersecurity All rights reserved'.</p>	
<p>- So we just need to not use angular bracket, and the name for our lab is also encode it! so why not encode our angular brackets using URL encoding, "<" => "%3c" ">" => "%3e"</p>	
 <p>The screenshot shows the same 'Search and Filter' page. The search input field now contains the URL-encoded string <code>%3ch1%3eNext Try%3c/h1%3e</code>, which is highlighted with a red box. The 'Search' button is still present. The text below the input field says 'Your Searched results for'.</p>	

- The response to our new payload:



Vulnerable URLs

- https://labs.hacktify.in/HTML/html_lab/lab_6/html_injection_6.php

Consequences of not Fixing the Issue

- Unsafe URLs: Malicious links can be inserted into the webpage, exposing users to various forms of attacks, such as phishing scams or drive-by downloads.
- False positives: Blacklisted items might be legitimate in certain scenarios, leading to false positives where valid content gets rejected.
- Circumvention: Determined attackers can find ways around blacklists, either by obfuscating the malicious content or finding alternative techniques.

Suggested Countermeasures

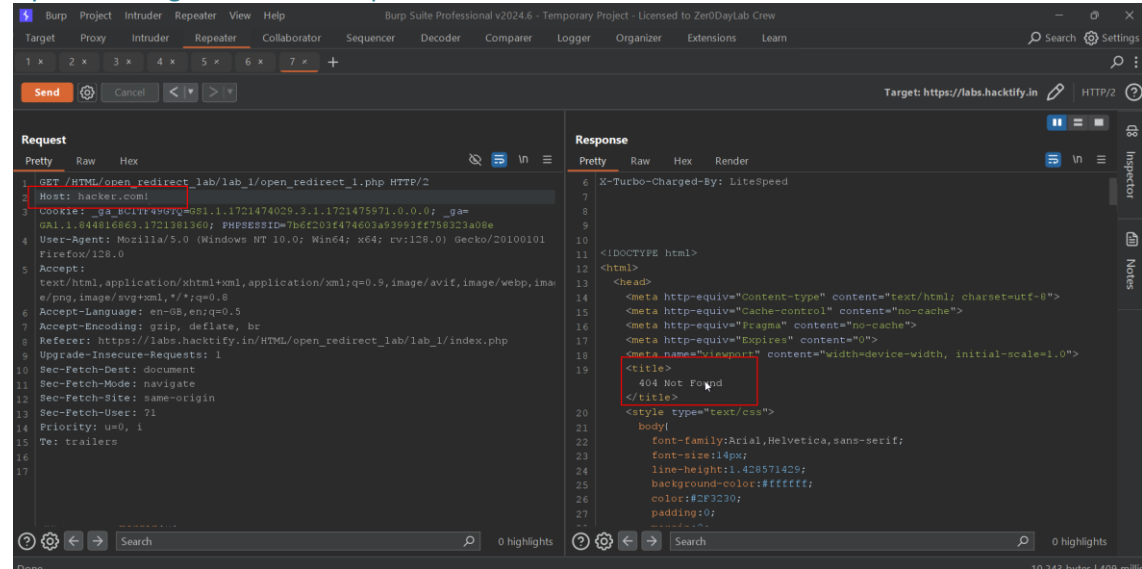
- Avoid overreliance on blacklisting and combine it with other security measures.
- Use libraries and frameworks that have built-in security features to prevent HTML injection attacks.
- Use Content Security Policy (CSP) to restrict the types of content that can be loaded on the webpage.

References

- <https://www.invicti.com/learn/html-injection/>

2. Open Redirect

2.1. A Simple Host

Reference	Risk Rating
A Simple Host	Low
Tools Used	
Burp Suite.	
Vulnerability Description	
It is redirecting to the specified link	
How It Was Discovered	
By monitoring the URLs in Burp Suite.	
	
Vulnerable URLs	
https://labs.hacktify.in/HTML/open_redirect_lab/lab_1/open_redirect_1.php	
Consequences of not Fixing the Issue	
Phishing Attacks: <ul style="list-style-type: none">• Deceptive URLs: Attackers can exploit open redirects to create deceptive URLs that appear to be legitimate but actually redirect users to malicious sites.• Credential Theft: Users can be tricked into entering sensitive information (e.g., usernames, passwords, credit card details) on malicious websites.	
Impact on Search Engine Ranking: <ul style="list-style-type: none">• SEO Penalties: Search engines may penalize websites involved in redirecting to malicious content, affecting their search rankings and visibility.	
Suggested Countermeasures	
Server-Side Validation: <ul style="list-style-type: none">• Validate redirect targets on the server side before performing the redirect. Ensure the target is a trusted destination.	
References	
https://portswigger.net/kb/issues/00500100_open-redirection-reflected	

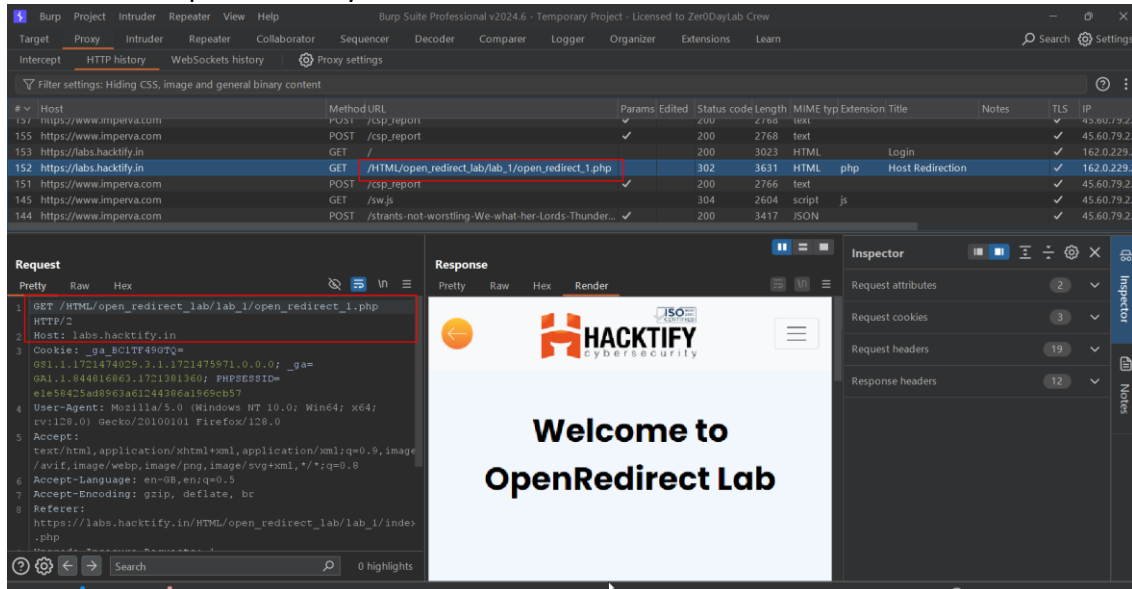
Proof of Concept

Step 1 –

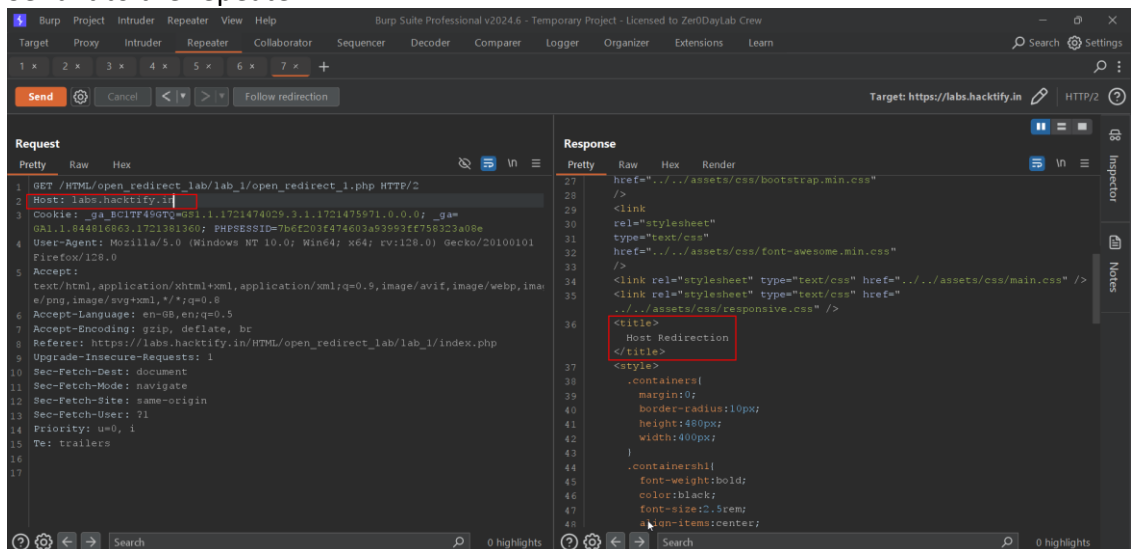
Click on Start lab

Step 2 –

Check the Burp HTTP Proxy.

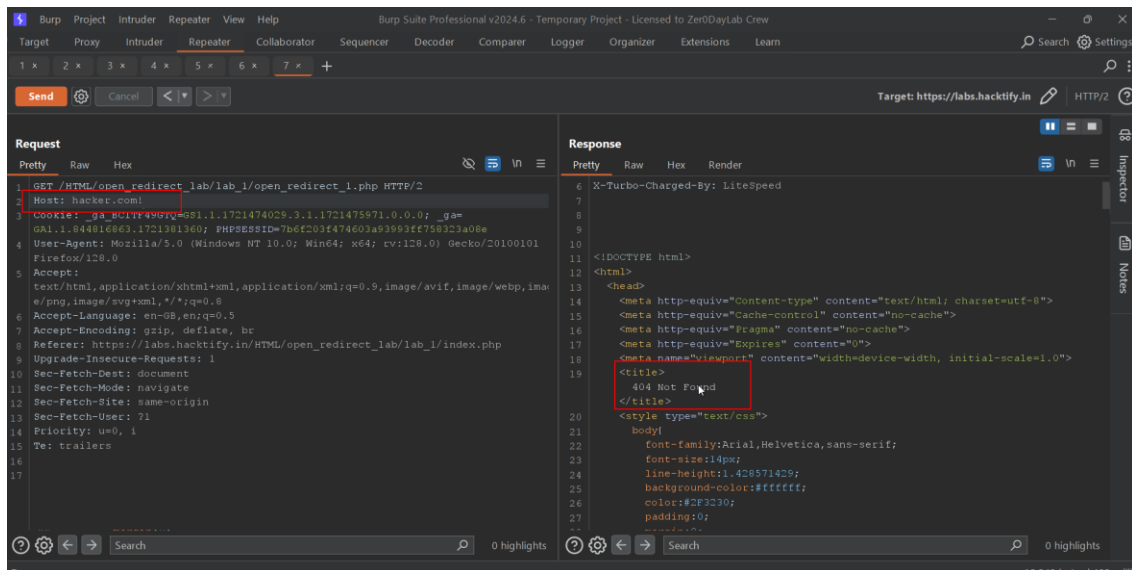


Send it to the repeater



Step 3 –

Send it to the repeater and change the host name



We can observe that instead of going to `open_redirect_lab/lab_1/open_redirect_1.php` we are getting redirected to `https://labs.hacktify.in/`

2.2. Story of a beautiful header

Reference	Risk Rating
Story of a beautiful header	Low
Tools Used	
Burp Suite.	
Vulnerability Description	
It is redirecting to the specified link	
How It Was Discovered	
<p>The screenshot shows the Burp Suite interface with a request and response view. The request is a GET to <code>/HTML/open_redirect_lab/lab_2/open_redirect_2.php</code>. The response is a 302 Found from <code>https://labs.hacktify.in/</code>. The response body shows an HTML document with a title "302 Found" and some CSS styling.</p>	
Vulnerable URLs	
<code>https://labs.hacktify.in/HTML/ open_redirect_lab/lab_2/open_redirect_2.php</code>	

Consequences of not Fixing the Issue
<ul style="list-style-type: none"> The X-Forwarded-Host header can be manipulated by attackers to perform various attacks, such as open redirect, SSRF (Server-Side Request Forgery), and bypassing security controls.
Suggested Countermeasures
<p>Strict Header Validation:</p> <ul style="list-style-type: none"> Whitelist Valid Hosts: Only allow known and trusted hostnames and domains. Reject any requests with X-Forwarded-Host values that are not in the whitelist. Regex Validation: Use regular expressions to validate the format of the X-Forwarded-Host header against expected patterns. <p>2. Server-Side Configuration:</p> <ul style="list-style-type: none"> Proxy Configuration: Configure your reverse proxy to remove or sanitize the X-Forwarded-Host header before passing the request to the backend server. Disable Header Trust: If not required, disable trust in the X-Forwarded-Host header by configuring the web server or application to ignore it. <p>3. Use Secure Alternatives:</p> <ul style="list-style-type: none"> Forwarded Header: Consider using the standard Forwarded header instead of X-Forwarded-Host as it is more structured and can include multiple pieces of information (by, for, host, proto). <p>4. Application Logic:</p> <ul style="list-style-type: none"> Context-Aware Logic: Ensure that the application logic that processes or uses the X-Forwarded-Host header is context-aware and checks for anomalies or unexpected values. Sanitization: Sanitize and escape any values derived from the X-Forwarded-Host header before using them in responses or logs. <p>5. Logging and Monitoring:</p> <ul style="list-style-type: none"> Log Abnormal Values: Log any abnormal or unexpected values in the X-Forwarded-Host header for further analysis and alerting. Monitor Requests: Use monitoring tools to detect and alert on unusual patterns or spikes in requests that include the X-Forwarded-Host header. <p>6. Security Headers:</p> <ul style="list-style-type: none"> Content Security Policy (CSP): Implement a CSP that restricts the sources of content and URLs that can be loaded by the browser. HTTP Strict Transport Security (HSTS): Use HSTS to ensure that all communications with the server are made over HTTPS, reducing the risk of man-in-the-middle attacks.
References

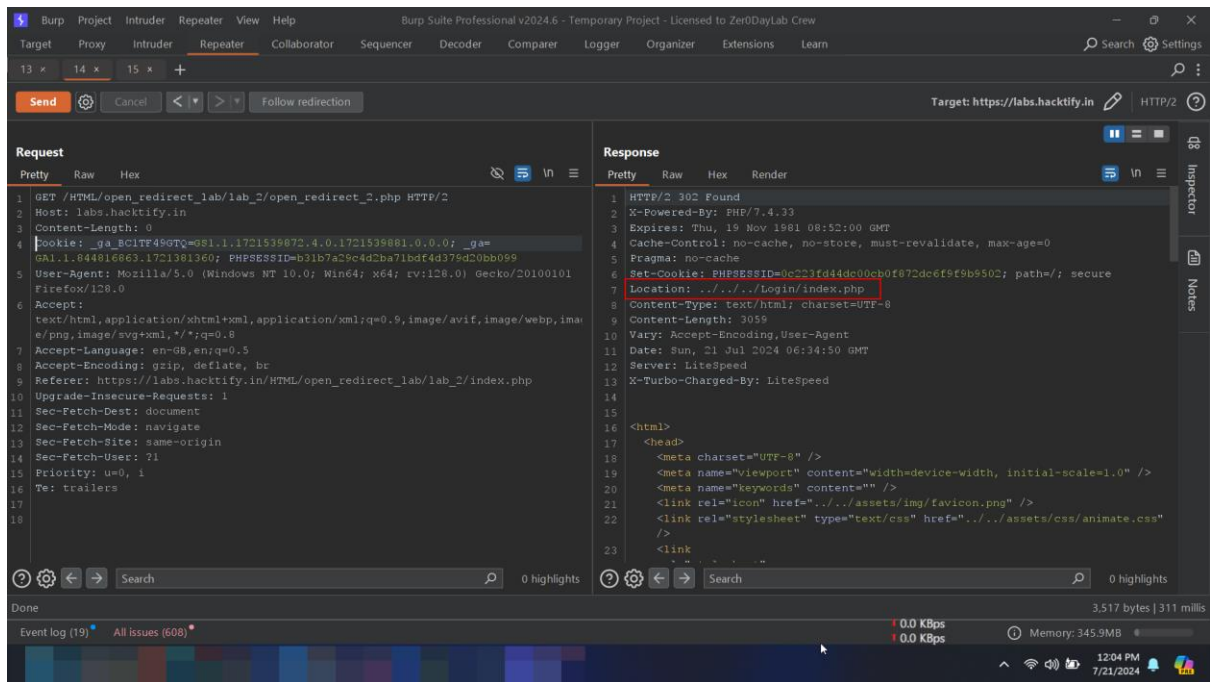
Proof of Concept

Step 1 –

Click on Start lab and check the HTTP Proxy in Burp Server.

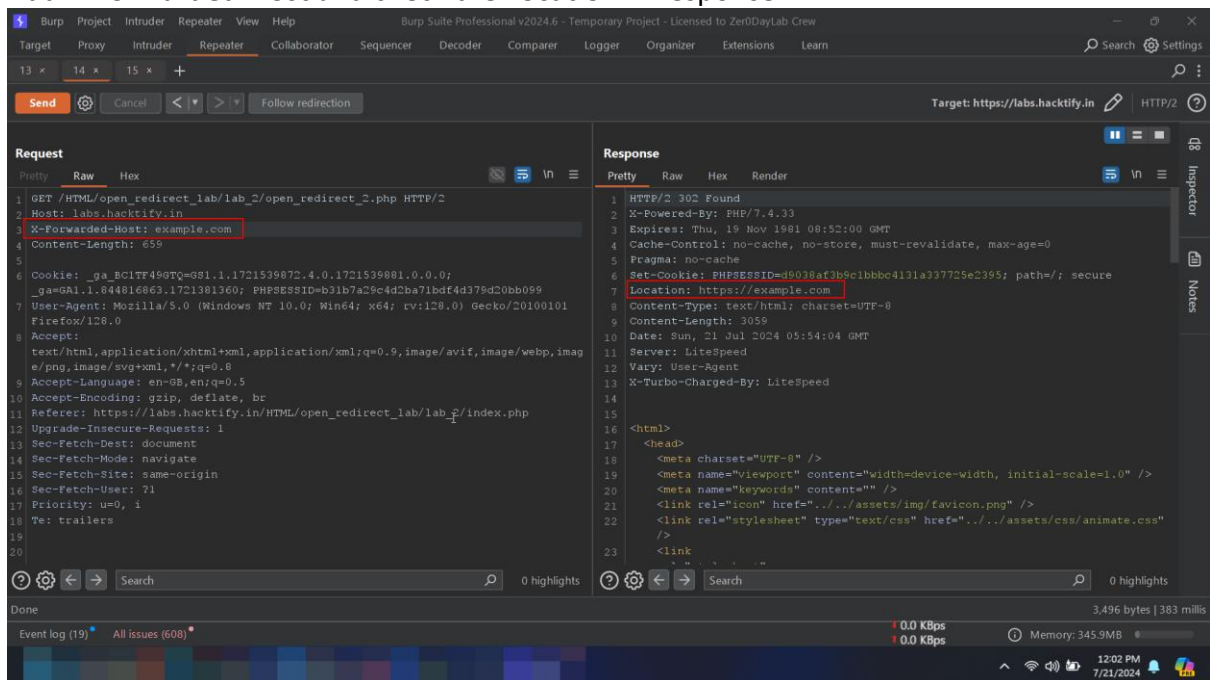
Step 2 –

Send it to the repeater.



Step 3 –

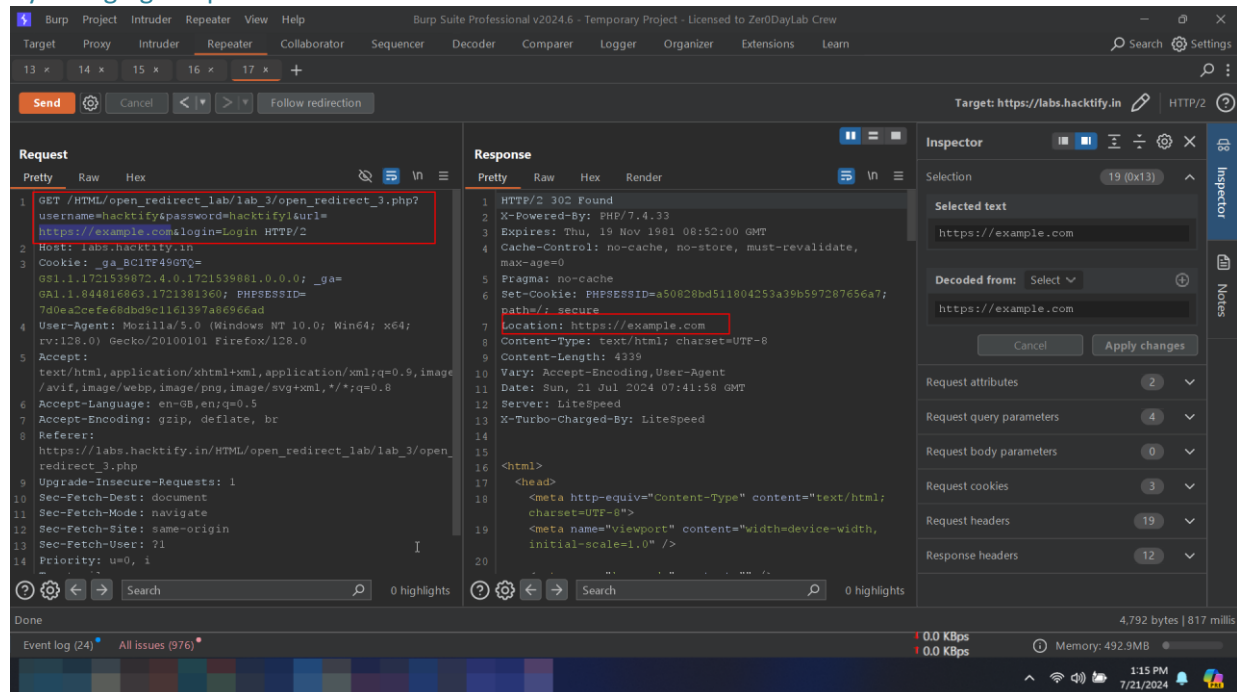
Add X-Forwarded-Host and check the Location in response.



2.3. Sanitize Params

Reference	Risk Rating
Sanitize Params	Medium
Tools Used	
Burp Suite.	
Vulnerability Description	
It is redirecting to the specified link	
How It Was Discovered	

By changing the parameter's value.



Vulnerable URLs

https://labs.hacktify.in/HTML/open_redirect_lab/lab_3/open_redirect_3.php

Consequences of not Fixing the Issue

- Failure to properly sanitize parameters can lead to several serious consequences, including security breaches, user deception, and overall system compromise.

Suggested Countermeasures

Sanitization and Validation:

- **Whitelist Approach:** Implement a whitelist of allowed URLs or paths to which redirects are permitted.
- **Parameter Validation:** Ensure that redirect parameters are strictly validated against expected patterns or values.

References

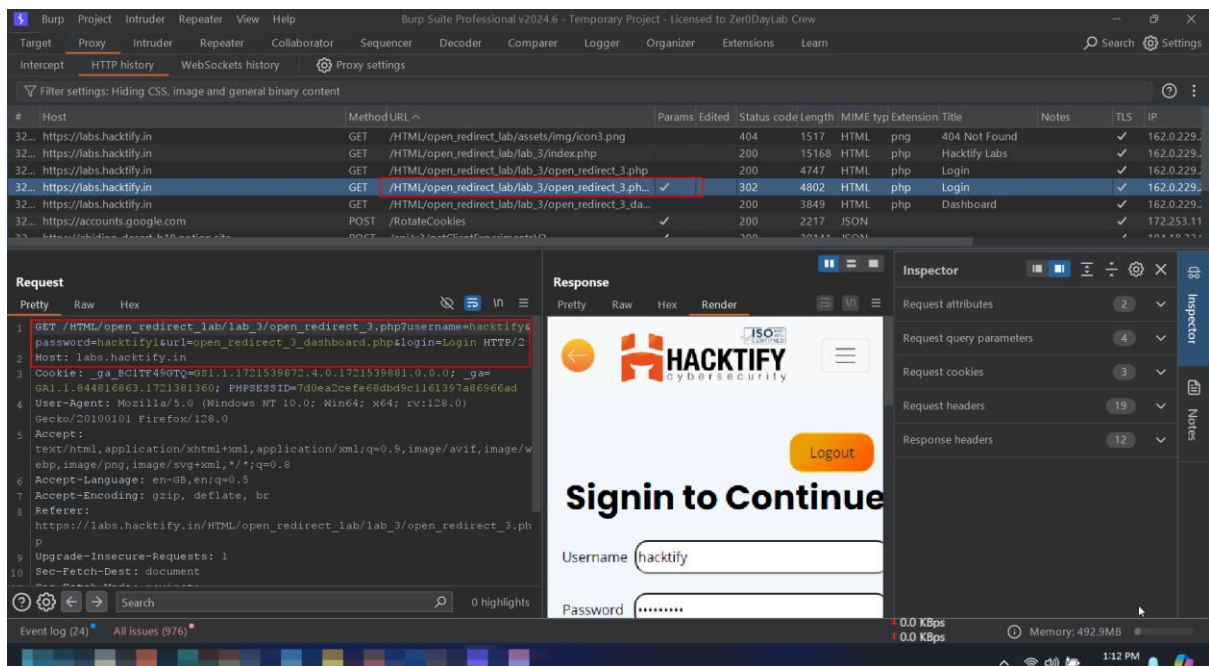
Proof of Concept

Step 1 –

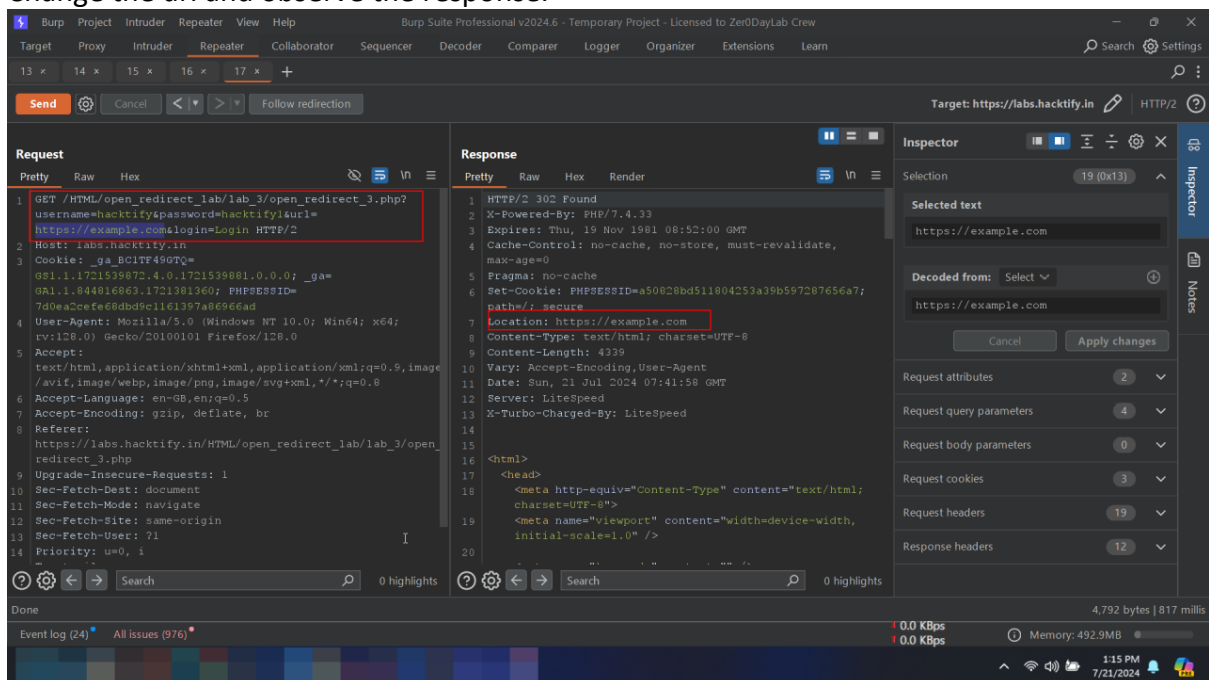
Click on Start lab

Step 2 –

Check the Burp HTTP Proxy.



Step 3 –
Change the url and observe the response.



2.4. Patterns are Important

Reference	Risk Rating
Patterns are Important	Medium
Tools Used	
Burp Suite.	
Vulnerability Description	
It is redirecting to the specified link	

How It Was Discovered

1. Initial Interpretation:

- When a URL like `open_redirect_4_dashboard.php///example.com&login=Login` is processed by a web server, the part after `open_redirect_4_dashboard.php` (i.e., `///example.com&login=Login`) can be misinterpreted as a path or parameter to be handled by the script.

2. Path Confusion:

- The `///` in the URL can be interpreted by the web server as the start of an absolute URL. Some servers or scripts might interpret this as an attempt to reference `http://example.com`.

3. Parameter Handling:

- If the script `open_redirect_4_dashboard.php` is designed to handle redirects based on user input (e.g., a next parameter), it might extract `example.com` as the target host and `login=Login` as an additional parameter.

Breakdown of the URL

- Script Name:** `open_redirect_4_dashboard.php`
- Path and Parameters:** `///example.com&login=Login`

The screenshot displays the Burp Suite Professional interface. The 'Request' tab on the left shows a GET request to `/HTML/open_redirect_lab/lab_4/open_redirect_4.php?username=hacktify&password=hacktify&login=open_redirect_4_dashboard.php///example.com&login=Login HTTP/2`. The 'Response' tab in the center shows a 302 Found status with a Location header pointing to `https://example.com`. The 'Inspector' tab on the right highlights the selected text `///example.com` and shows the decoded location `https://example.com`.

Vulnerable URLs

`https://labs.hacktify.in/HTML/ open_redirect_lab/lab_4/open_redirect_4.php`

Consequences of not Fixing the Issue

- The attacker can use the stolen credentials to access sensitive information, perform fraudulent transactions, or further exploit the compromised accounts.
- Users who realize they have been phished may lose trust in the legitimate site, resulting in reputational damage and potential loss of business for the company.
- The company may face legal and regulatory repercussions for failing to protect user data.

Suggested Countermeasures

- Strict Validation:**

Validate and sanitize all input parameters. Ensure only allowed URLs are used for redirects. Use a whitelist of allowed redirect URLs.

- **Relative URLs:**
Prefer relative URLs over absolute URLs for redirects within the application.
- **Regular Expressions:**
Use regular expressions to ensure the input conforms to expected patterns.
- **Explicit Redirects:**
Use explicit mapping for known routes instead of user-supplied input.

References

<https://brightsec.com/blog/open-redirect-vulnerabilities/>

2.5. File Upload!? Redirect it!

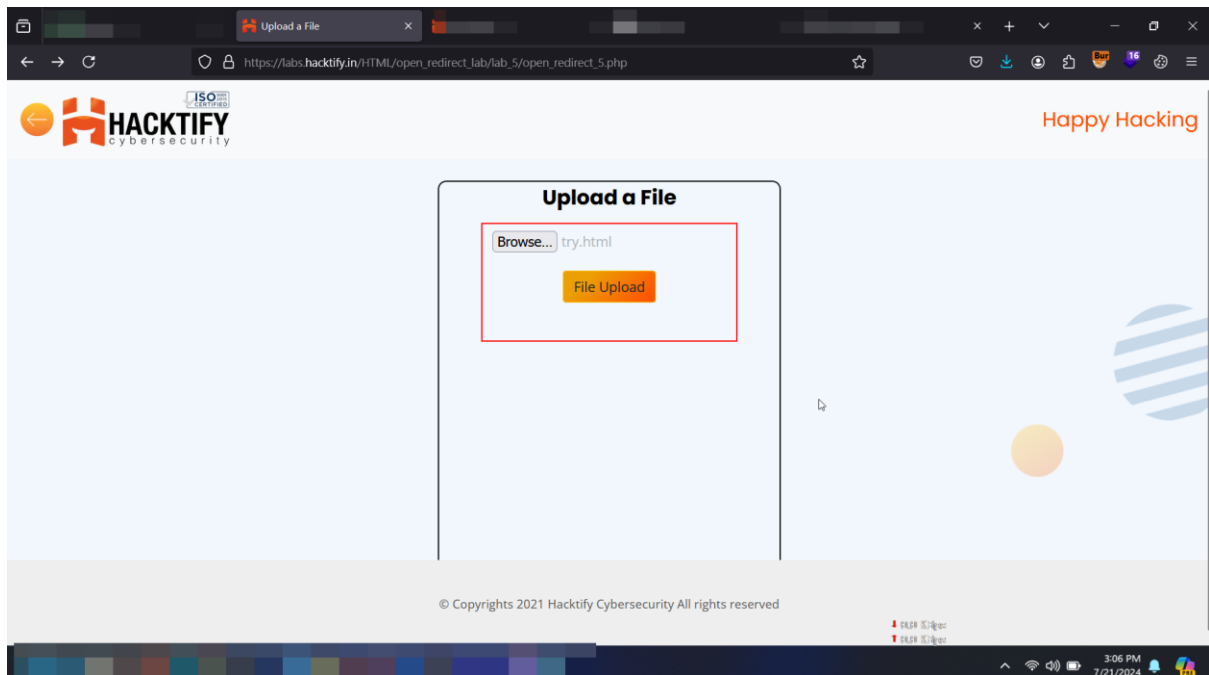
Reference	Risk Rating
File upload!? Redirect it!	Low
Tools Used	
Burp Suite.	
Vulnerability Description	
It is redirecting to the specified link	
How It Was Discovered	
By uploading the file having targeted location	
Vulnerable URLs	
https://labs.hacktify.in/HTML/open_redirect_lab/lab_5/open_redirect_5.php	
Consequences of not Fixing the Issue	
<ol style="list-style-type: none">1. Phishing Attacks:<ul style="list-style-type: none">○ User Deception: Redirects to malicious sites can trick users into disclosing sensitive information.○ Brand Damage: Users lose trust in the compromised site.2. Malware Distribution:<ul style="list-style-type: none">○ Drive-by Downloads: Users are redirected to sites that automatically download malware.	

<ul style="list-style-type: none"> ○ Widespread Infection: Malicious software spreads, compromising multiple systems.
3. Security Bypass: <ul style="list-style-type: none"> ○ Circumventing Security Measures: Redirects can bypass web application firewalls and other security filters. ○ Chained Exploits: Combined with other vulnerabilities, they enable more complex attacks.
4. Data Leakage: <ul style="list-style-type: none"> ○ Exposure of Sensitive Information: Sensitive data can be inadvertently exposed to unauthorized parties. ○ Unintended Data Sharing: Session tokens and other sensitive parameters can be leaked.
5. Loss of User Trust: <ul style="list-style-type: none"> ○ Erosion of Confidence: Users lose trust in the site, potentially leading to attrition.
6. Legal and Regulatory Consequences: <ul style="list-style-type: none"> ○ Compliance Violations: Breaches can result in fines and legal penalties. ○ Legal Liability: Organizations may face lawsuits from affected users.
Suggested Countermeasures
1. Validate and Sanitize File Uploads: <ul style="list-style-type: none"> ○ Whitelist File Types: Only allow safe file types (e.g., images). ○ Content Inspection: Check files for malicious content.
2. Secure Storage and Access: <ul style="list-style-type: none"> ○ Store Files Outside Web Root: Prevent direct access to uploaded files. ○ Serve Files Securely: Use secure methods to serve files.
3. URL Validation: <ul style="list-style-type: none"> ○ Sanitize URLs: Ensure URLs do not contain redirects. ○ Whitelist Redirects: Only allow redirects to trusted URLs.
4. Content Security Policy (CSP): <ul style="list-style-type: none"> ○ Implement CSP Headers: Restrict navigations and redirects to trusted origins.
References https://portswigger.net/web-security/file-upload

Proof of Concept

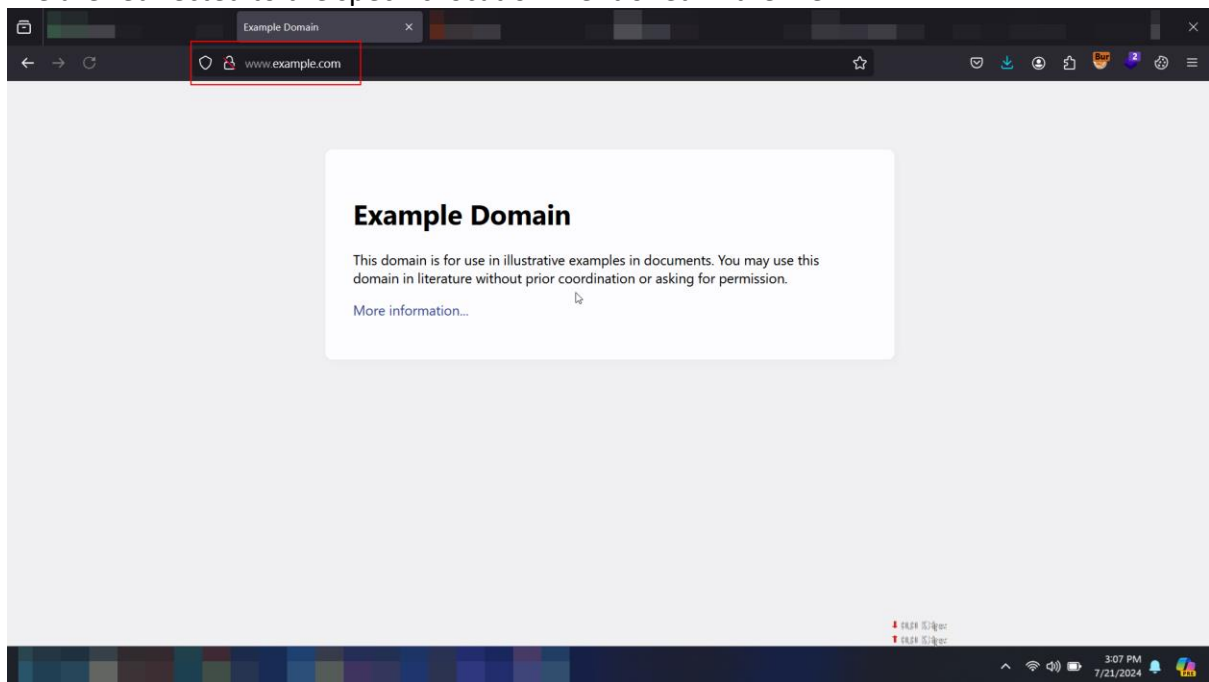
Step 1 –

Click on Start lab, select the file to upload



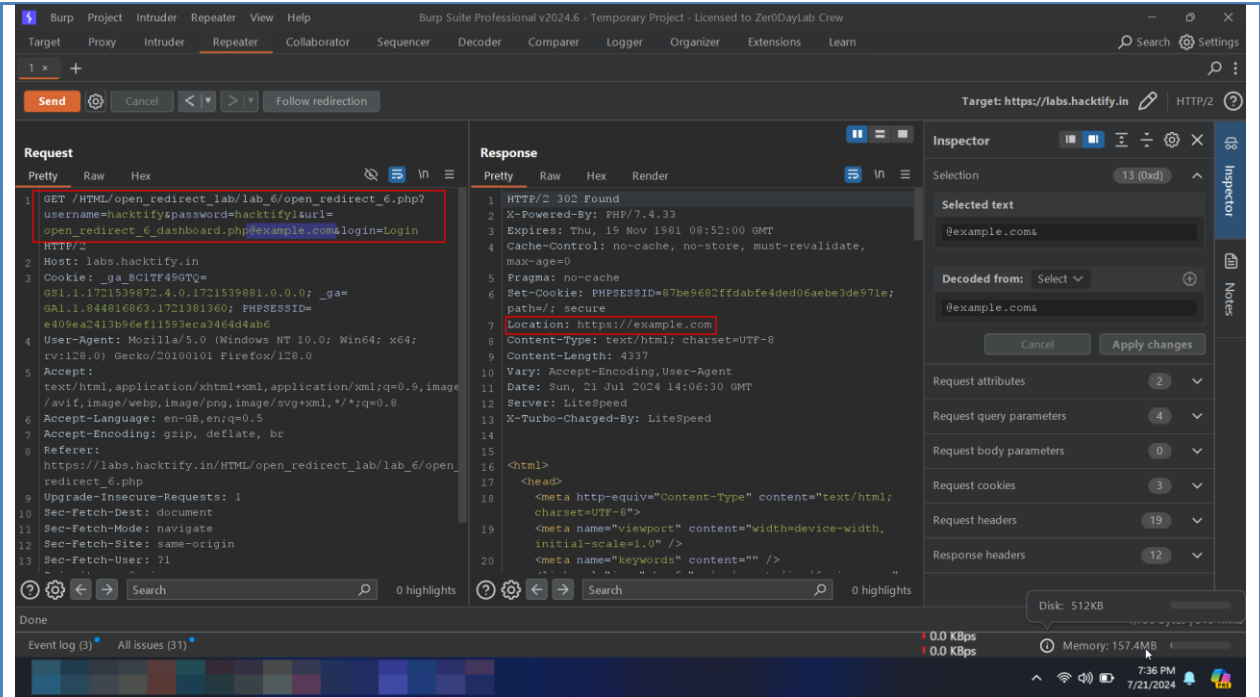
Step 2 –

We are redirected to the specific location mentioned in the file



2.6. Same Param Twice!

Reference	Risk Rating
Same Param Twice!	Hard
Tools Used	
Burp Suite.	
Vulnerability Description	
It is redirecting to the specified link	
How It Was Discovered	
By adding @oururl after given URL.	



Vulnerable URLs

https://labs.hacktify.in/HTML/ open_redirect_lab/lab_6/open_redirect_6.php

Consequences of not Fixing the Issue

- Exploiting open redirect vulnerabilities can lead to serious consequences, including phishing attacks that trick users into providing sensitive information, malware distribution that infects user devices, combined attacks with XSS vulnerabilities, manipulation of search engine rankings through SEO poisoning, reputation damage to the legitimate site, theft of session tokens or other sensitive data, and potential denial-of-service (DoS) attacks by overwhelming target sites with redirected traffic.

Suggested Countermeasures

- Validate and Sanitize Inputs:** Only allow redirects to trusted URLs or domains.
- Use Relative URLs:** Avoid absolute URLs to prevent user-controlled redirection.
- Implement Parameterized Redirects:** Map parameters to predefined URLs server-side.
- Monitor and Log Redirects:** Track redirection patterns to detect abuse.
- User Education:** Inform users about the risks of clicking on suspicious links.

References

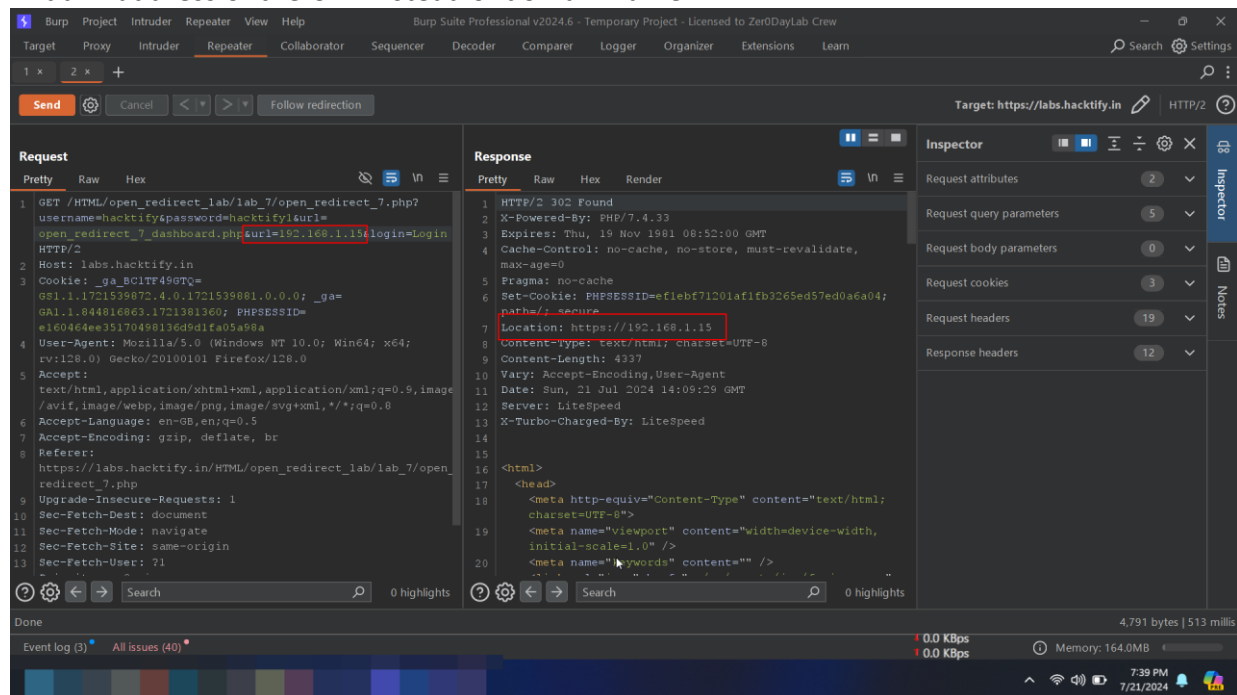
https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html

2.7. Domains? Not Always!

Reference	Risk Rating
Domains? Not Always!	Hard
Tools Used	
Burp Suite.	
Vulnerability Description	
It is redirecting to the specified link	

How It Was Discovered

. Add IP address of the URL instead of domain name.



Vulnerable URLs

https://labs.hacktify.in/HTML/open_redirect_lab/lab_7/open_redirect_7.php

Consequences of not Fixing the Issue

- Open redirects can lead to phishing attacks and malware distribution, compromising user security. They also undermine trust, facilitate affiliate fraud, and can be exploited for SEO abuse.

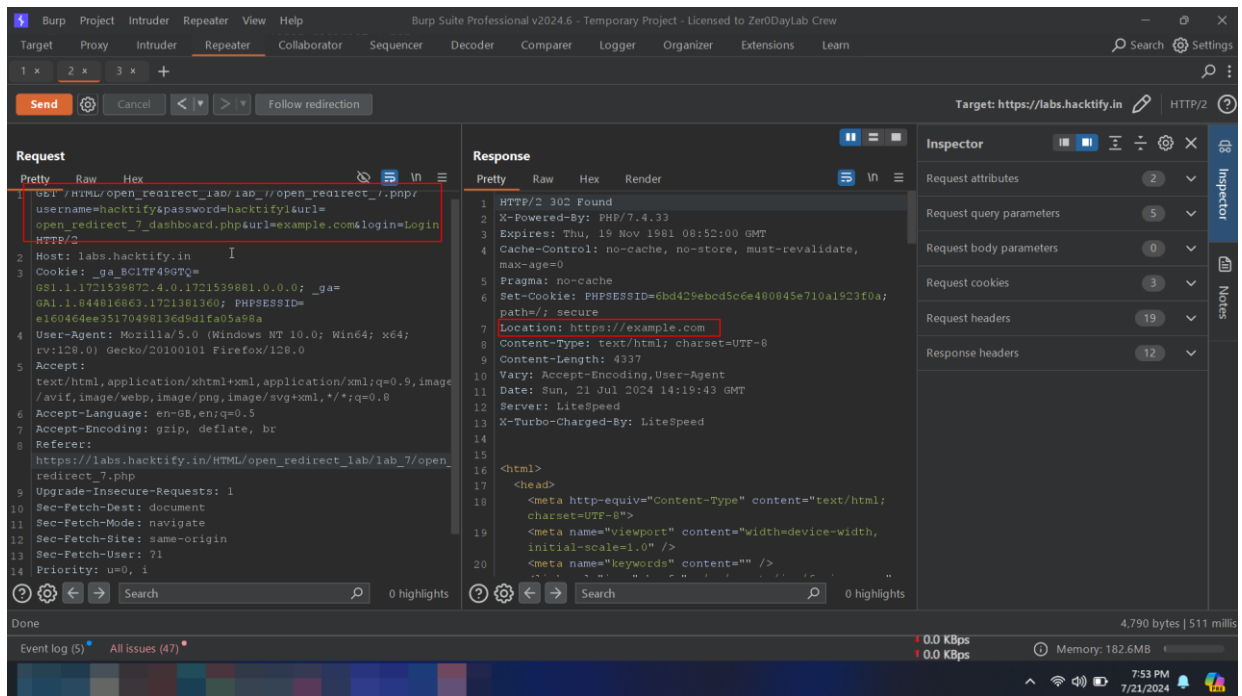
Suggested Countermeasures

- validating and sanitizing all user inputs that influence redirection URLs, ensuring they only allow redirects to trusted domains.

References

OR

Add one more URL parameter.



2.8. Style Digit Symbol

Reference	Risk Rating
Style Digit Symbol	Hard
Tools Used	
Burp Suite.	
Vulnerability Description	
It is redirecting to the specified link	
How It Was Discovered	
Add IP address of the URL instead of domain name.	

Vulnerable URLs

https://labs.hacktify.in/HTML/open_redirect_lab/lab_1/open_redirect_1.php

Consequences of not Fixing the Issue

- Open redirects can lead to phishing attacks and malware distribution, compromising user security. They also undermine trust, facilitate affiliate fraud, and can be exploited for SEO abuse.

Suggested Countermeasures

- validating and sanitizing all user inputs that influence redirection URLs, ensuring they only allow redirects to trusted domains.

References

https://portswigger.net/kb/issues/00500100_open-redirection-reflected