**Week 2**

**Penetration Testing Report**

**Introduction**

This report document hereby describes the proceedings and results of a Black Box security assessment conducted against the **Week 2 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 2 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, clickjacking** |
| --- | --- |

**3. Summary**

Outlined is a Black Box Application Security assessment for the **Week 2 Labs**.

**Total number of Sub-labs: 8 Sub-labs**

| **High** | **Medium** | **Low** |
| --- | --- | --- |
| **1** | **3** | **4** |

**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**

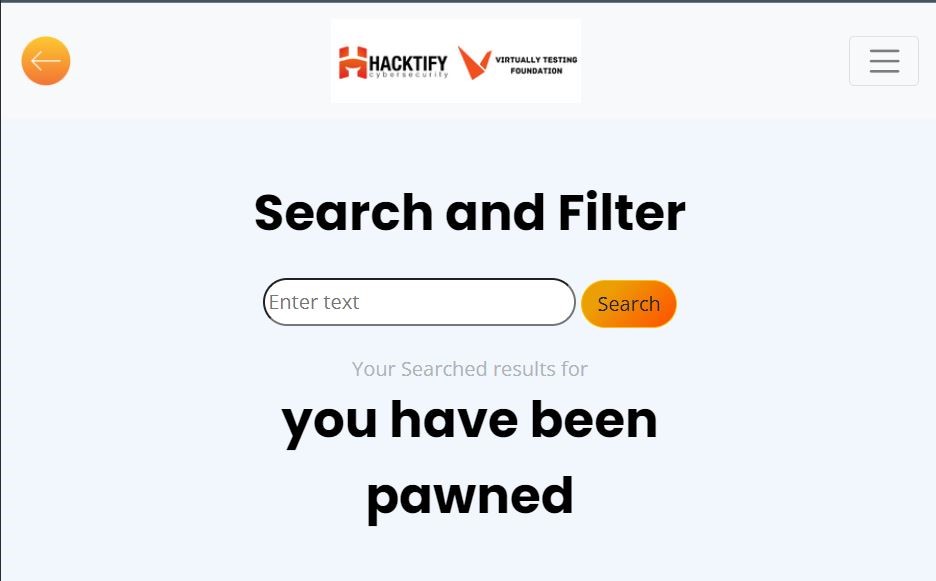
# 1. HTML injection

# 1.1. HTML’s Are Easy

| **Reference** | **Risk Rating** |
| --- | --- |
| HTML’s Are Easy | **Low** |
| **Tools Used** | |
| by manual injection | |
| **Vulnerability Description** | |
| It is a basic html injection vulnerability in the search bar if we enter the payload in that search bar it will reflected to the same page as browser search result | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://www.bugbountyhunter.org/internship_labs/HTML/html_lab/lab_1/html_injection_1.php> | |
| **Consequences of not Fixing the Issue** | |
| It can allow an attacker to modify the page by injecting a payload. It can affect only the current user. | |
| **Suggested Countermeasures** | |
| Do not allow parsing or execution of HTML tags from the user input.  Every input should be checked if it contains any script code or any HTML code. One should check, if the code contains any special script or HTML brackets – <script></script>, <html></html> | |
| **References** | |
| <https://www.softwaretestinghelp.com/html-injection-tutorial/>  <https://www.imperva.com/learn/application-security/html-injection/> | |

# Proof of Concept

payload : <h1>you have been pawned</h1>



# 

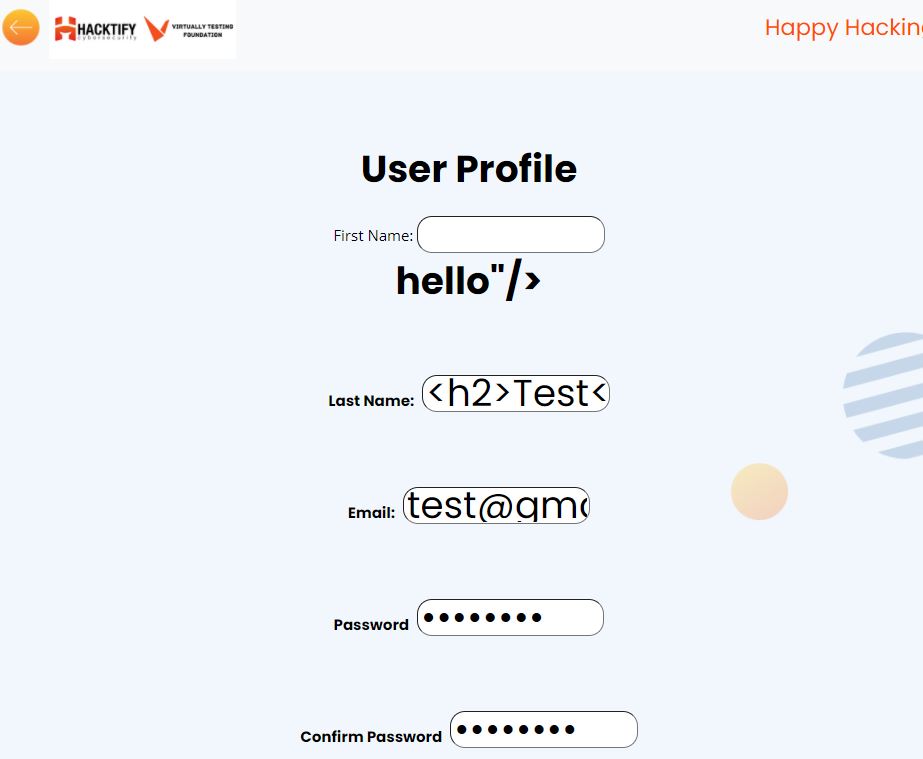
# 

# 1.2. Let Me Store Them!

| **Reference** | **Risk Rating** |
| --- | --- |
| Let Me Store Them! | **Medium** |
| **Tools Used** | |
| by manual injection | |
| **Vulnerability Description** | |
| Stored HTML Injection: stored injection attack occurs when malicious HTML code is saved in the web server and is being executed every time when the user calls an appropriate functionality.\  stored html injection vulnerability in register page on first name field and last name field if we enter the payload in first name field bar it will stored and reflected in that particular page. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://www.bugbountyhunter.org/internship_labs/HTML/html_lab/lab_2/profile.php> | |
| **Consequences of not Fixing the Issue** | |
| when the payload is executed it is stored in the server permanently whenever the user login to that website it will got executed | |
| **Suggested Countermeasures** | |
| don’t accept html tags in input fields  Use appropriate validation for input and output | |
| **References** | |
| <https://hardik-solanki.medium.com/html-injection-stored-which-ultimately-resulted-into-a-cve-2020-26049-61c1a47dc2e8> | |

# Proof of Concept

payload:<h1>hello<hello>



# 

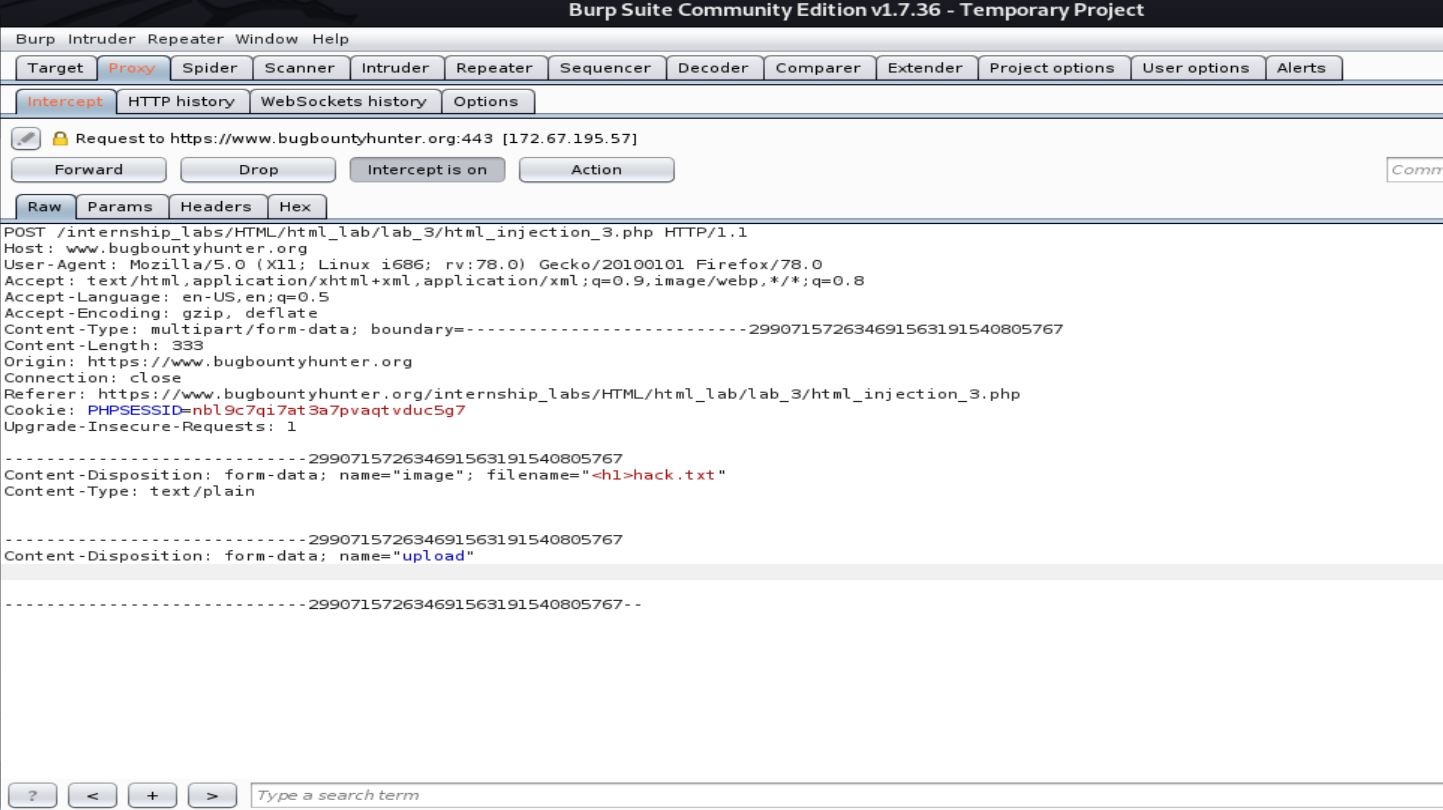
# 1.3. File Names Are Also Vulnerable!

| **Reference** | **Risk Rating** |
| --- | --- |
| File Names Are Also Vulnerable! | **Low** |
| **Tools Used** | |
| Burp Suite | |
| **Vulnerability Description** | |
| In this attack an attacker can upload a malicious file with a malicious file name and he can able to execute that file . | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://www.bugbountyhunter.org/internship_labs/HTML/html_lab/lab_3/html_injection_3.php> | |
| **Consequences of not Fixing the Issue** | |
| It may affect the current user | |
| **Suggested Countermeasures** | |
| stop allowing html tags and don’t allow <,>./ etc this type of tags  file name uploaded in the website should be properly validated and don’t allow these type of special characters (<,/,>...etc) and it should be properly validated. | |
| **References** | |
| <https://www.vistainfosec.com/blog/comprehensive-guide-on-html-injection/> | |

# 

# Proof of Concept

**payload injected to file name:<h1>hack.txt**





# 

\

# 1.4. File Content And HTML Injection A Perfect Pair!

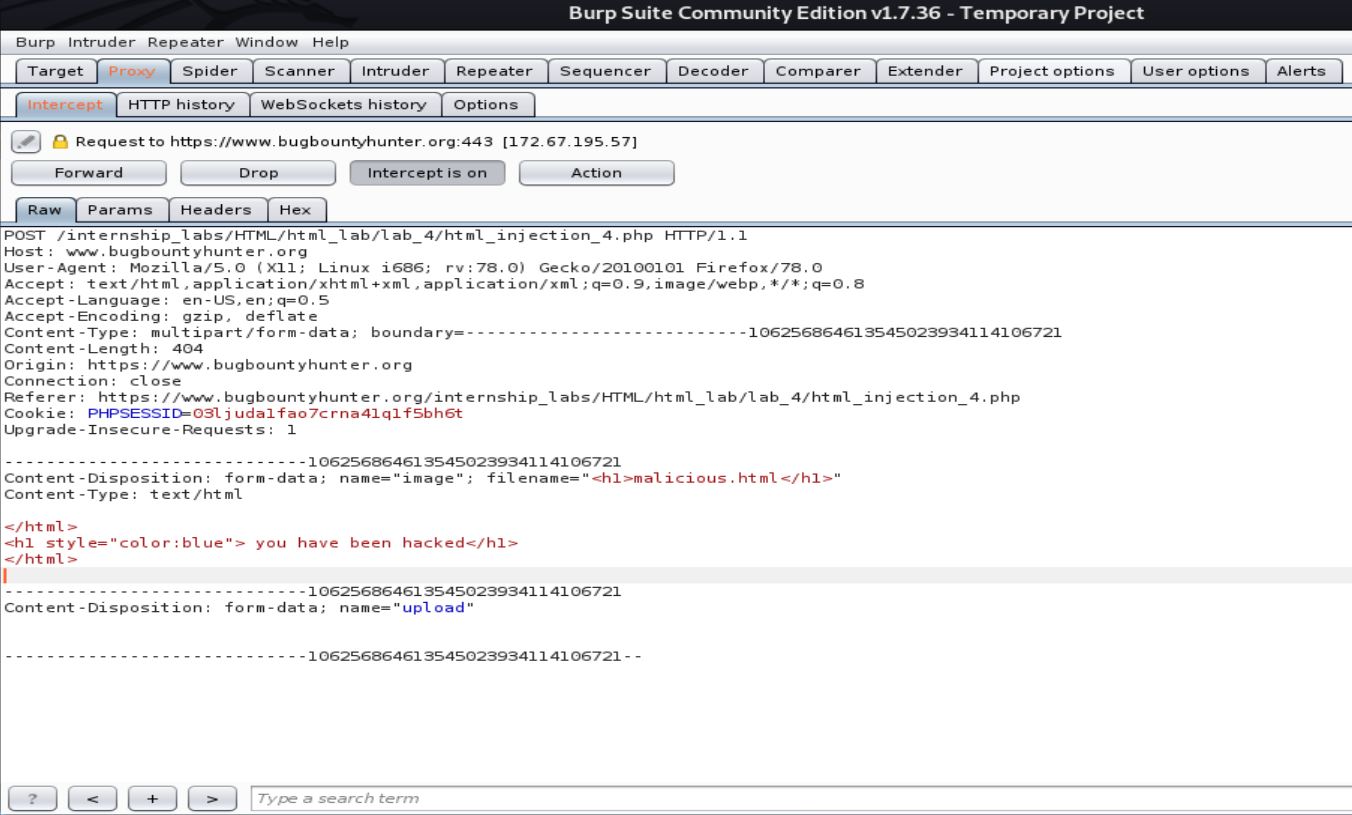
| **Reference** | **Risk Rating** |
| --- | --- |
| File Content And HTML Injection A Perfect Pair! | **Medium** |
| **Tools Used** | |
| Burp Suite | |
| **Vulnerability Description** | |
| In this attack an attacker can upload a file with some malicious script inside the file once the file is uploaded the script gets executed . | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://www.bugbountyhunter.org/internship_labs/HTML/html_lab/lab_4/html_injection_4.php> | |
| **Consequences of not Fixing the Issue** | |
| Attacker can perform any action on the web page and the method is post method it affects only the current user and the attacker can able to change the webpage content. | |
| **Suggested Countermeasures** | |
| file name uploaded in the website should be properly validated and don’t allow these type of special characters (<,/,>...etc) and it should be properly validated.  file must be validated properly during file upload.  A developer must restrict the output result of html files  processing of html files in file uploading session must be denied | |
| **References** | |
| <https://blog.yeswehack.com/yeswerhackers/exploitation/file-upload-attacks-part-1/> | |

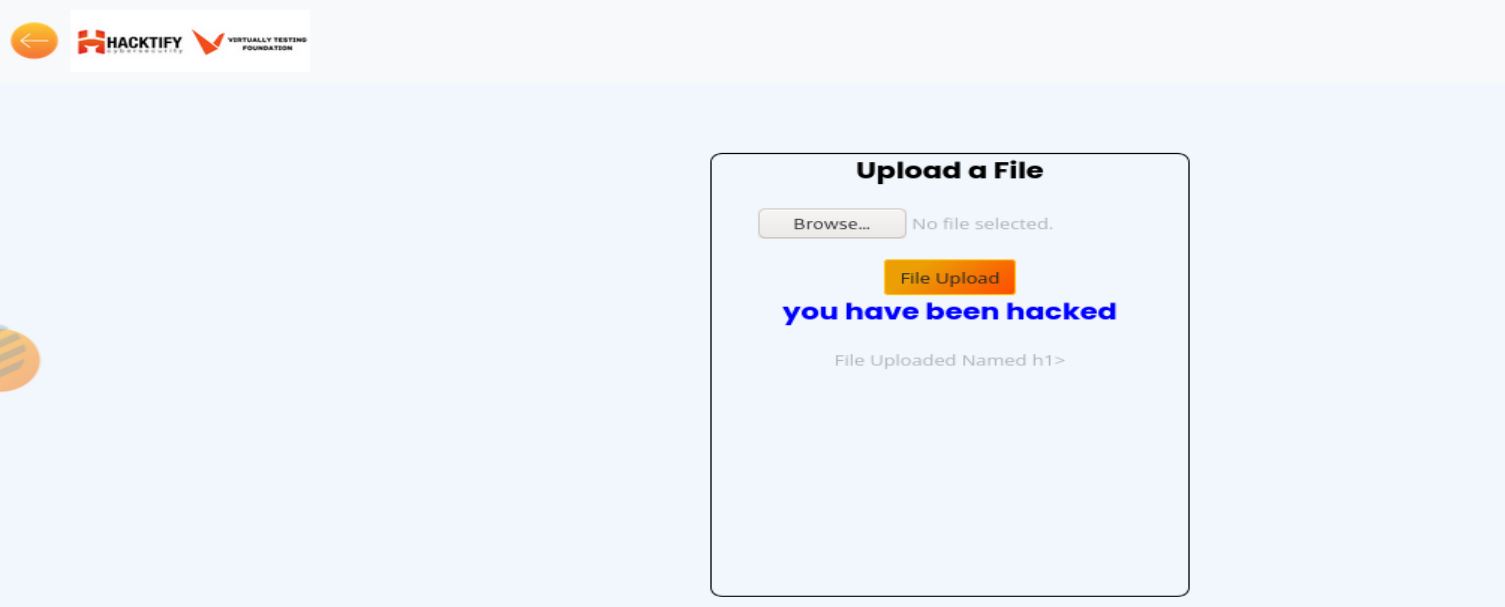
# 

# Proof of Concept

payload injected by file name :<h1>malicious.html</h1>

inside the html file :<h1 style=”color:blue”> u=you have been hacked</h1>





# 

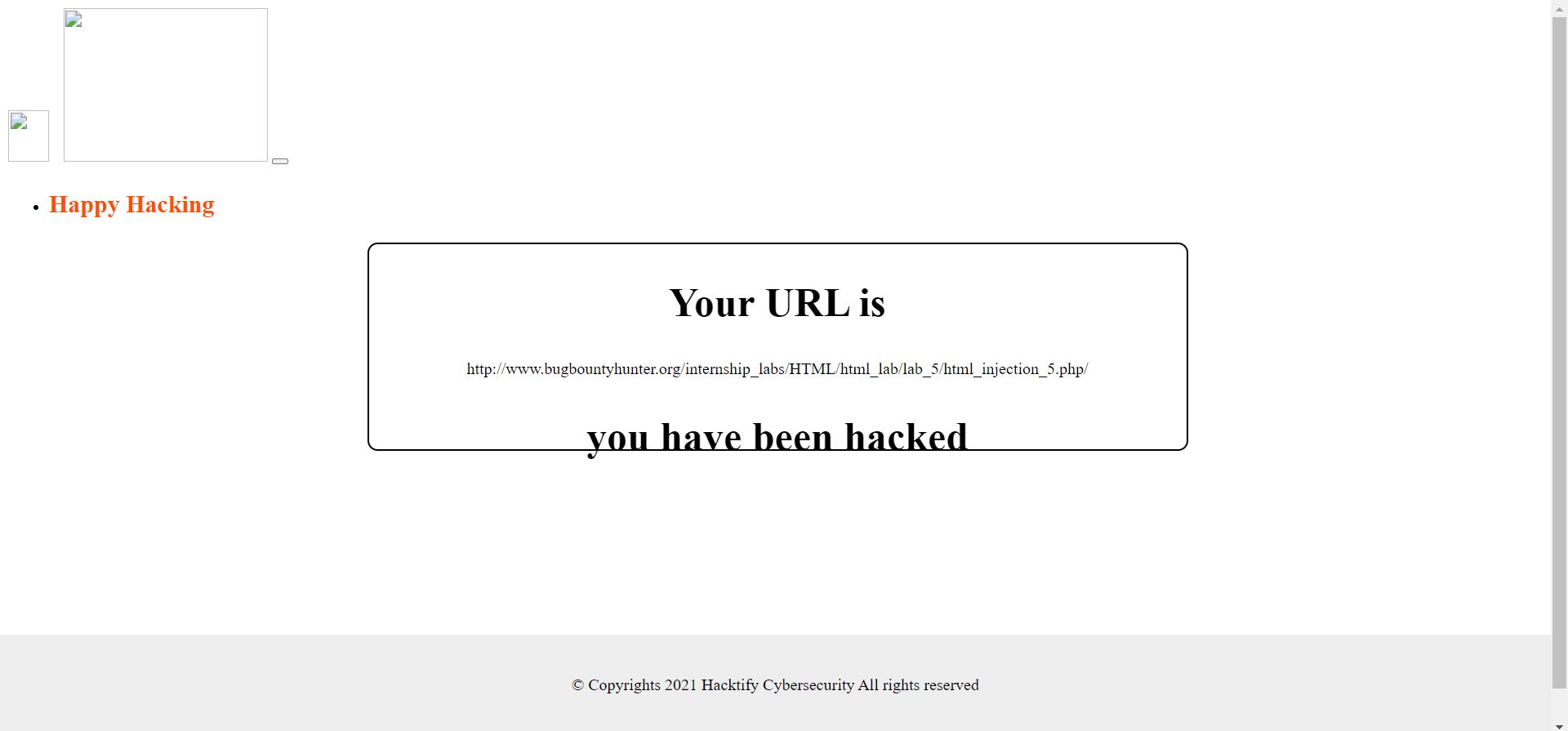
# 1.5. Injecting HTML Using URL

| **Reference** | **Risk Rating** |
| --- | --- |
| Injecting HTML Using URL | **High** |
| **Tools Used** | |
| by manual injection | |
| **Vulnerability Description** | |
| An attacker can send a payload injected url to victim once the victim click and visit the url and user enter his credentials it may lead to phishing | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| <https://www.bugbountyhunter.org/internship_labs/HTML/html_lab/lab_5/html_injection_5.php> | |
| **Consequences of not Fixing the Issue** | |
| it may lead to account take over by sending payload injected url to victim  Attacker can perform any action on the web page and can also create it as a phishing page to  divert all users to other attacker controlled web page. | |
| **Suggested Countermeasures** | |
| don’t accept the html tags in url fields and restrict those tags in url fields | |
| **References** | |
| <https://hackerone.com/reports/111915> | |

# 

# Proof of Concept

payload:[https://www.bugbountyhunter.org/internship\_labs/HTML/html\_lab/lab\_5/html\_injection\_5.php/<h1>you have been hacked</h1>](https://www.bugbountyhunter.org/internship_labs/HTML/html_lab/lab_5/html_injection_5.php/%3ch1%3eyou%20have%20been%20hacked%3c/h1%3e)



# 

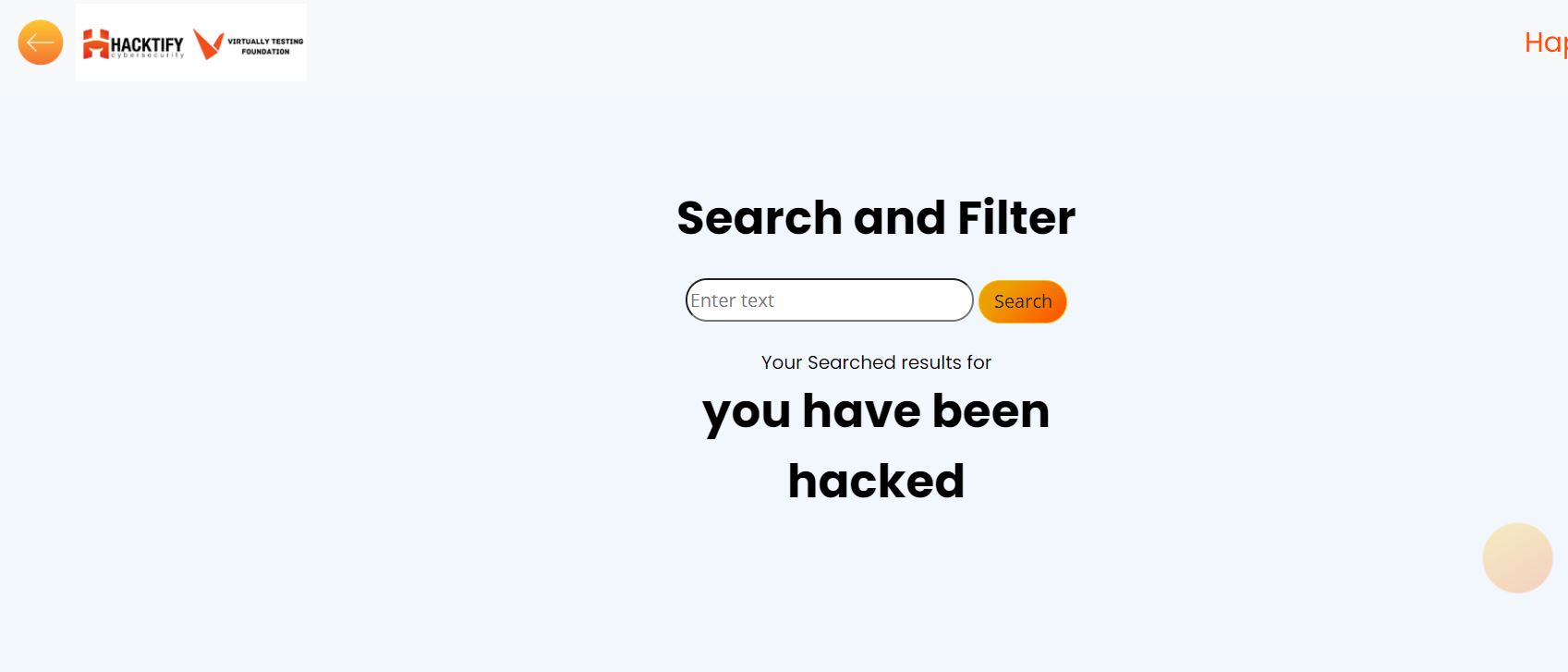
# 1.6. Encode IT!

| **Reference** | **Risk Rating** |
| --- | --- |
| Encode IT! | **Low** |
| **Tools Used** | |
| <https://meyerweb.com/eric/tools/dencoder/> | |
| **Vulnerability Description** | |
| In this vulnerability an attacker can able to inject a html payload as encoded format.It affects only the current user | |
| **How It Was Discovered** | |
| Manual analysis | |
| **Vulnerable URLs** | |
| <https://www.bugbountyhunter.org/internship_labs/HTML/html_lab/lab_6/html_injection_6.php> | |
| **Consequences of not Fixing the Issue** | |
| It affects only the current user when the page gets refreshed it comes to original form.And the risk rating is low. | |
| **Suggested Countermeasures** | |
| don’t allow these type of special characters (<,/,>...etc)  don’t accept html tag as encoded format in the search field | |
| **References** | |
| <https://owasp.org/www-project-web-security-testing-guide/latest/6-Appendix/D-Encoded_Injection> | |

# 

# Proof of Concept

payload: %3Ch1%3Eyou%20have%20been%20hacked%3C%2Fh1%3E



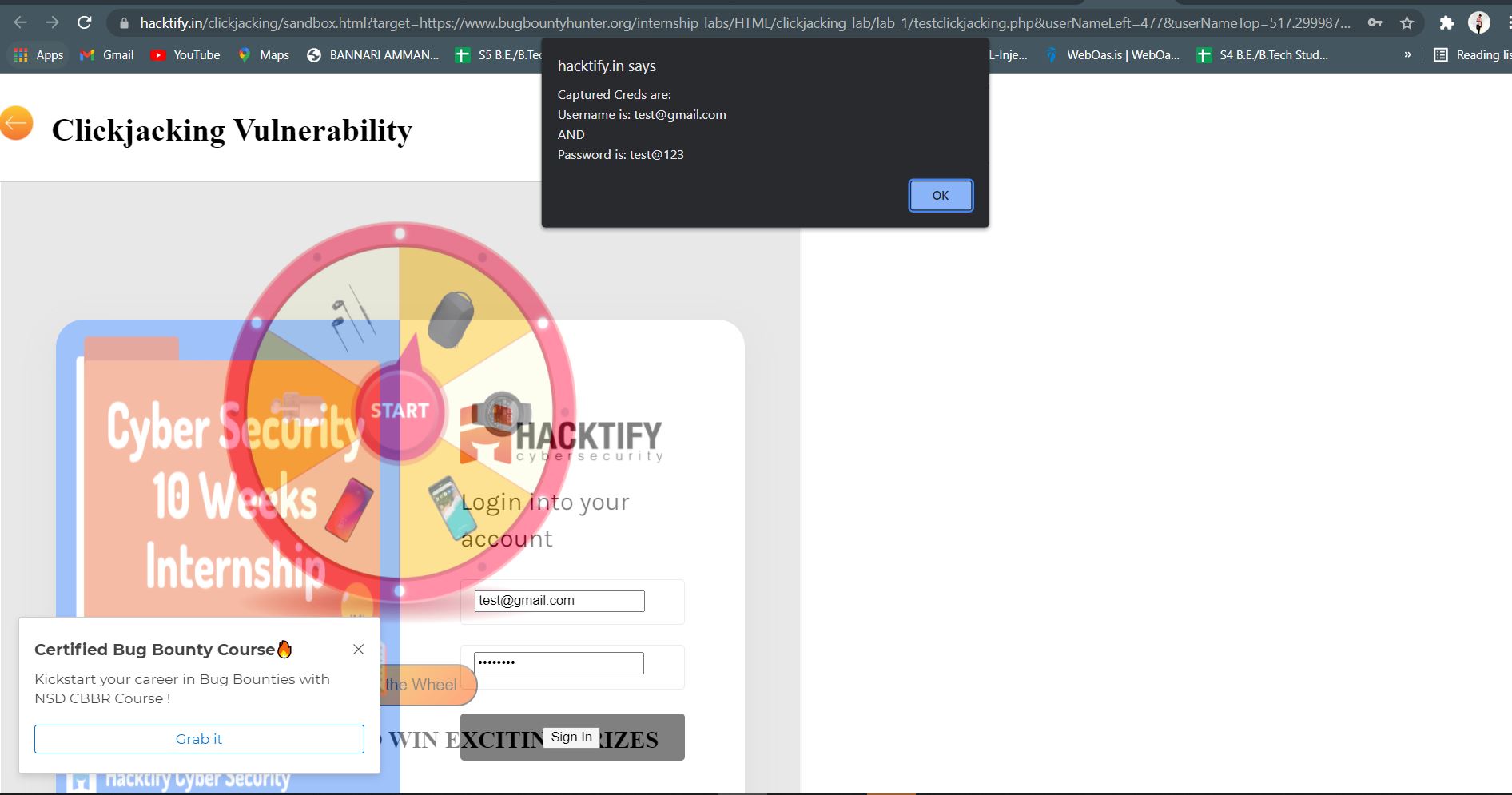
# 2. Clickjacking

# 2.1. Let's Hijack!

| **Reference** | **Risk Rating** |
| --- | --- |
| Let's Hijack! | **High** |
| **Tools Used** | |
| <https://hacktify.in/clickjacking/> | |
| **Vulnerability Description** | |
| Clickjacking is an attack that tricks a user into clicking a webpage element which is invisible or disguised as another element. This can cause users to unwittingly download malware, visit malicious web pages, provide credentials or sensitive information, transfer money, or purchase products online | |
| **How It Was Discovered** | |
| Automated tool | |
| **Vulnerable URLs** | |
| <https://www.bugbountyhunter.org/internship_labs/HTML/clickjacking_lab/lab_1/testclickjacking.php> | |
| **Consequences of not Fixing the Issue** | |
| it can steal the credentials of the user | |
| **Suggested Countermeasures** | |
| Sending the proper Content Security Policy (CSP) frame-ancestors directive response headers that instruct the browser to not allow framing from other domains. (This replaces the older X-Frame-Options HTTP headers.)  Employing defensive code in the UI to ensure that the current frame is the most top level window | |
| **References** | |
| <https://portswigger.net/web-security/clickjacking> | |

# 

# Proof of Concept



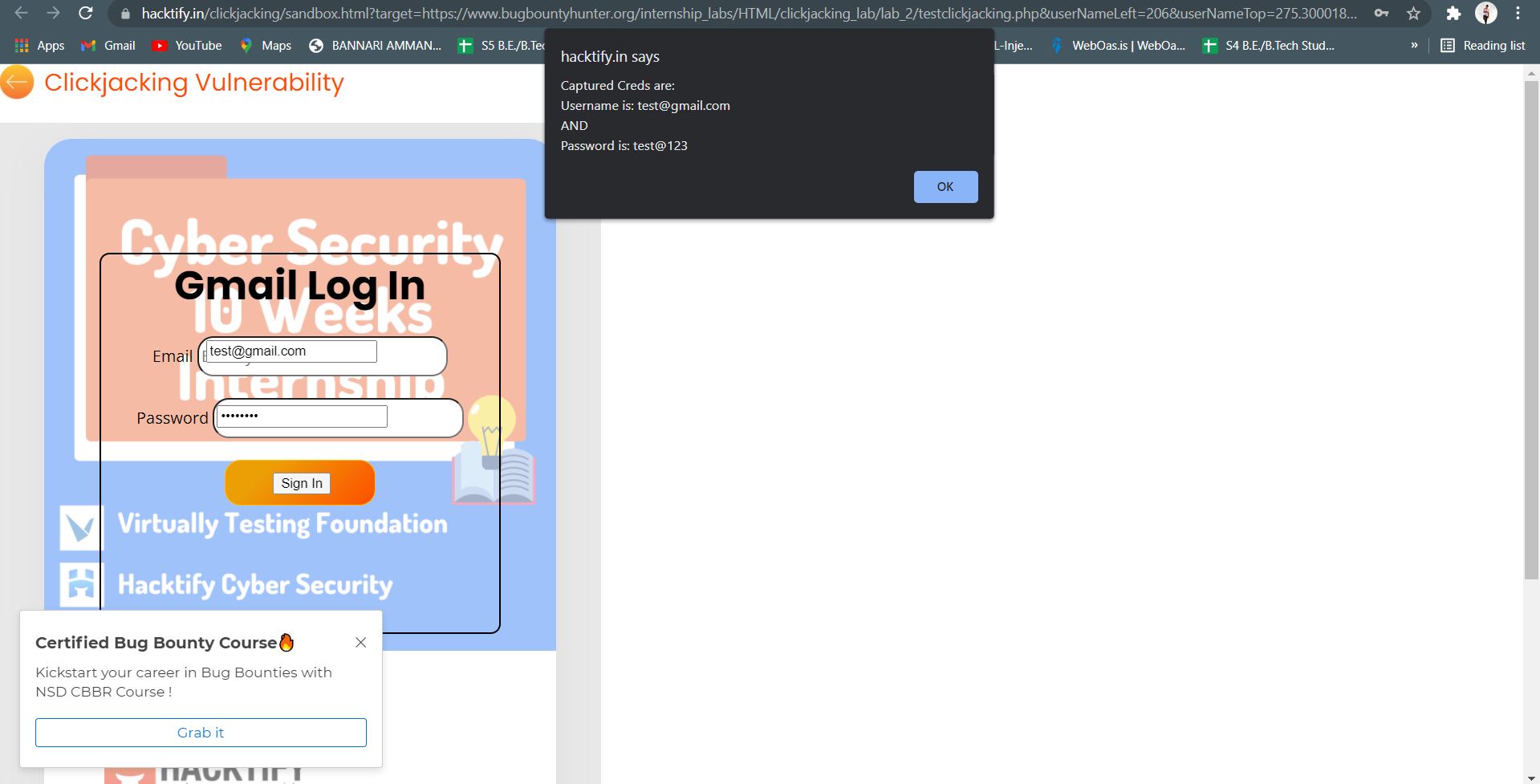
# 

# 2.2. Re-Hijack!

| **Reference** | **Risk Rating** |
| --- | --- |
| Re-Hijack! | **High** |
| **Tools Used** | |
| <https://hacktify.in/clickjacking/> | |
| **Vulnerability Description** | |
| Clickjacking is an attack that tricks a user into clicking a webpage element which is invisible or disguised as another element. This can cause users to unwittingly download malware, visit malicious web pages, provide credentials or sensitive information, transfer money, or purchase products online | |
| **How It Was Discovered** | |
| Automated tool | |
| **Vulnerable URLs** | |
| <https://www.bugbountyhunter.org/internship_labs/HTML/clickjacking_lab/lab_2/testclickjacking.php> | |
| **Consequences of not Fixing the Issue** | |
| It can steal the credentials of the user | |
| **Suggested Countermeasures** | |
| Add a framekiller to the website: Javascript has a framekiller function that stops pages from being pulled into an iFrame | |
| **References** | |
| <https://owasp.org/www-community/attacks/Clickjacking>  <https://portswigger.net/web-security/clickjacking> | |

# 

# Proof of Concept

****