

## IE2062 – Web Security

Year 2, Semester 2

### TryHackMe Room



BSc (Hons) in Information Technology

Sri Lanka Institute of Information Technology

Malabe

Sri Lanka

Student Name	Registration Number
P T D Minipura	IT23298408

## Table of Contents

Room Overview .....	3
1. Title: Cracking Weak CSP – Misconfigurations in Action.....	3
2. Key Concepts .....	3
Learning Objectives.....	4
Room Structure.....	5
1. Task 1 – Understanding Content Security Policy (CSP) .....	6
2. Task 2 – Importance of Content Security Policy (CSP) .....	7
3. Task 3 – Understanding CSP for Web Security .....	9
4. Task 4 - Understanding the Limitations of Content Security Policy (CSP).....	11
5. Task 5 - Hand on CSP digging .....	13
6. Task 6 - CSP Bypass — Inline Event Handler Challenge.....	15
7. Task 7 – [Vulnerable Web App].....	18
Room Link .....	21
Reflection .....	22
1. Personal Insights and Lessons Learned While Designing the Room .....	22
2. How This Room Contributes to the Learning Community.....	22
Workings.....	24
Create Virtual Machine.....	24
Ubuntu Server Installation .....	26
Implementation of Challenges .....	33
Downloaded File Challenge 1 .....	37
Downloaded File Challenge 2.....	38
References.....	40

# Room Overview

## 1. Title: Cracking Weak CSP – Misconfigurations in Action

This room focuses on real-world misconfigurations in Content Security Policies (CSP) that fail to effectively safeguard web applications against Cross-Site Scripting (XSS) attacks. Learners will explore how poorly implemented CSP rules, rather than providing protection, can create a false sense of security and leave applications vulnerable. The room highlights common weaknesses such as the use of overly permissive directives like 'unsafe-inline', allowing the execution of malicious inline scripts. Participants will also encounter scenarios where CSP headers are missing crucial directives, improperly scoped or misused with wildcards, leading to policy bypasses. Through hands-on challenges, learners will practice analyzing flawed CSP configurations, crafting payloads that exploit these weaknesses and understanding how attackers can inject and execute arbitrary JavaScript despite the presence of CSP. By the end of the room, participants will gain practical skills in identifying CSP issues and understanding best practices for implementing strong, effective CSPs that genuinely enhance the security posture of web applications.

## 2. Key Concepts

### 1. CSP Headers

- HTTP response headers that define which resources (scripts, styles, images, etc.) are allowed to load and execute.
- Help prevent attacks like Cross-Site Scripting (XSS) by enforcing strict resource loading rules.

### 2. Inline Scripts

- JavaScript code written directly in the HTML document inside <script> tags.
- Often a security risk, as attackers can exploit them if input validation is weak.
- CSP can control their execution by using directives like 'unsafe-inline', nonces or hashes.

### 3. Wildcards

- The \* character allows resources to be loaded from any origin.
- Simplifies configuration but can dangerously weaken security.
- Should be avoided in sensitive directives like script-src to maintain strong protection.

#### 4. Secure Policy Crafting

- Involves carefully specifying trusted sources and avoiding unsafe practices.
- Prefer nonces (nonce-...) or hashes over allowing inline scripts.
- Avoid wildcards and overly broad source definitions to minimize attack surfaces.
- Regularly audit and update policies as the application evolves.

## Learning Objectives

By completing this room, participants will develop a comprehensive set of skills and deepen their knowledge in the following areas:

### 1. Understanding CSP's Role in XSS Prevention:

Gain a clear understanding of how Content Security Policy (CSP) is designed to reduce the risk of Cross-Site Scripting (XSS) attacks by restricting the sources from which content can be loaded and executed.

### 2. Recognizing Misconfigured CSP Headers:

Learn to identify common mistakes in CSP implementation, such as overly permissive directives, missing critical restrictions or the improper use of wildcards and unsafe allowances.

### 3. Performing CSP Bypasses through Inline JavaScript and Input Injection:

Develop practical skills in exploiting weak CSP configurations by injecting malicious input or leveraging inline scripts to achieve script execution even when a CSP is present.

### 4. Applying Strong CSP Mitigations:

Acquire the knowledge to design and implement robust CSP policies, using techniques like nonces, hashes and strict source whitelisting, to effectively defend against XSS and related attacks.

## Room Structure

The following is the dashboard of my TryHackMe Room.

**Room Link:** <https://tryhackme.com/jr/crackingweakcspmisconfig>

The screenshot shows the TryHackMe room dashboard for the 'Cracking Weak CSP – Misconfigs' room. At the top, there's a navigation bar with links for Dashboard, Learn, Compete, Develop, and Other. On the right, there are buttons for Access Machines, a search bar, notifications, and a Go Premium button. Below the navigation is a breadcrumb trail: Learn > Cracking Weak CSP – Misconfigs. The main title is 'Cracking Weak CSP – Misconfigs' with a robot icon. A brief description states: 'This room introduces learners to misconfigured Content Security Policies that provide a false sense of security. Participants will analyze weak CSP headers, exploit bypasses using inline JavaScript, and learn best practices for secure CSP setup.' It indicates the room is Medium difficulty and takes 90 min. Below the description are buttons for Start AttackBox, Help, Save Room, and Options. A progress bar at the top right shows 'Room progress (95%)'. The central area has tabs for Chart, Scoreboard (which is selected), and Write-ups. The Scoreboard table shows the following data:

Rank	Username	[Task 1] #1	[Task 1] #2	[Task 1] #3	[Task 2] #1	[Task 2] #2	[Task 2] #3	[Task 2] #4	[Task 2] #5
1	Tharaka7 [0x1][NEOPHYTE]	✓	✓	✓	✓	✓	✓	✓	✓

Below the scoreboard is a section titled 'Room completed (100%)' which lists seven tasks with green checkmarks: Task 1 (Understanding Content Security Policy (CSP)), Task 2 (Importance of Content Security Policy (CSP)), Task 3 (Understanding CSP for Web Security), Task 4 ([Understanding the Limitations of Content Security Policy (CSP)]), Task 5 ([Hand on CSP digging]), Task 6 (CSP Bypass – Inline Event Handler Challenge), and Task 7 ([Vulnerable Web App]). At the bottom, there's a summary table with columns for Created by (Tharaka7), Room Type (Free Room. Anyone can deploy virtual machines in the room (without being subscribed)!), Users in Room (1), and Created (7 days ago).

## 1. Task 1 – Understanding Content Security Policy (CSP)

- This Room introduces Content Security Policy, a user can gain basic knowledge on this vulnerability.
- Users can read the content below and test themselves by answering the following questions.

Task 1 Understanding Content Security Policy (CSP)



**What is Content Security Policy?**

A Content Security Policy (CSP) is a security standard designed to add an additional layer of security for web applications. They allow developers to restrict which resources (such as JavaScript, CSS, Images and others) can be loaded. This helps reduce the risk of cross-site scripting (XSS), clickjacking, online skimming attacks such as Magecart and other code injection attacks. It is a defensive measure against any attacks that rely on executing malicious scripts and content from alternate domains in a trusted web context, or other attempts to circumvent the same-origin policy.

With CSP, you can limit which data sources are allowed by a web application, by defining the appropriate CSP directive in the HTTP response header.

Answer the questions below

What security standard helps restrict which resources (like JavaScript and CSS) can be loaded by a web application?

Content Security Policy

✓ Correct Answer

✗ Hint

Which type of attacks does Content Security Policy mainly help to prevent? (Name one)

-----

✗ Submit

✗ Hint

How is a Content Security Policy typically implemented in a web application?

-----

✗ Submit

✗ Hint

Answer the questions below

What security standard helps restrict which resources (like JavaScript and CSS) can be loaded by a web application?

Content Security Policy

✓ Correct Answer

✗ Hint

Which type of attacks does Content Security Policy mainly help to prevent? (Name one)

XSS (Cross-Site Scripting)

✓ Correct Answer

✗ Hint

How is a Content Security Policy typically implemented in a web application?

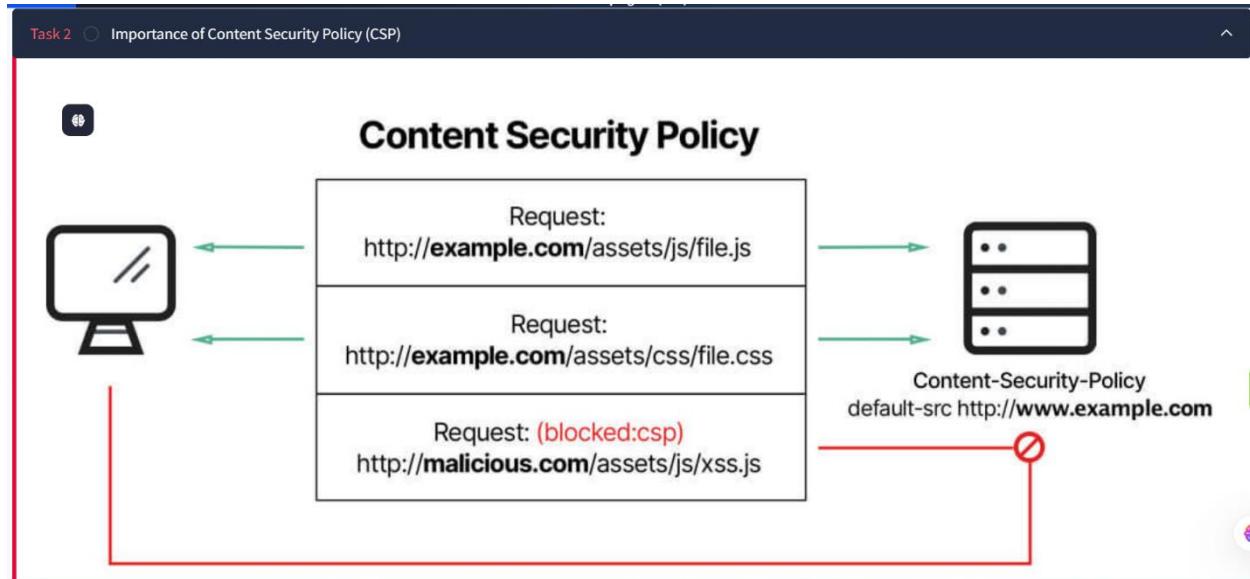
HTTP response header

✓ Correct Answer

✗ Hint

## 2. Task 2 – Importance of Content Security Policy (CSP)

- The importance of Content Security Policy is addressed here. This section understands the critical role of Content Security Policy (CSP) in mitigating Cross-Site Scripting (XSS) attacks and enforcing secure communication protocols like HTTPS.
- By answering the questions provided below the users can evaluate themselves.



### Why is a Content Security Policy Important?

#### Mitigating Cross-Site Scripting

The main purpose of CSP is to mitigate and detect XSS attacks. XSS attacks exploit the browser's trust in the content received from the server. The victim's browser is exposed to execution of malicious scripts, because the browser trusts the source of the content.

CSP allows server administrators to reduce or eliminate the ability of an attacker to trigger XSS, by specifying which Internet domains browsers should consider as legitimate sources of executable scripts. CSP-compliant browsers only run scripts contained in source files that are retrieved from whitelisted domains, and ignore all other scripts (including inline script and HTML event handling attributes).

#### Mitigating Packet Sniffing and Enforcing HTTPS

In addition to whitelisting domains from which a browser may load content, servers can also specify the allowed protocols. For example, the server can specify that browsers must load content via HTTPS.

A comprehensive data transfer protection policy involves not only implementing HTTPS in data transfer, but also marking all cookies with the secure attribute and automatically redirecting HTTP pages to HTTPS. Additionally, sites may use HTTP Strict-Transport-Security headers to ensure that browsers only connect to the site via encrypted channels.

Answer the questions below

What is the main type of web attack that CSP is designed to mitigate?

Submit

Hint

In CSP, what defines which domains are trusted to load scripts?

Submit

Hint

What protocol should a server enforce to protect data transfer using CSP?

Submit

Hint



Which HTTP header can a site use to force browsers to always connect securely?

Submit

Hint

Answer the questions below

Correct! Your answer is correct.

What is the main type of web attack that CSP is designed to mitigate?

Cross-Site Scripting (XSS)

✓ Correct Answer

Hint

In CSP, what defines which domains are trusted to load scripts?

Whitelisting domains

✓ Correct Answer

Hint

What protocol should a server enforce to protect data transfer using CSP?

HTTPS

✓ Correct Answer

Hint

Which HTTP header can a site use to force browsers to always connect securely?

HTTP Strict-Transport-Security

✓ Correct Answer

Hint

### 3. Task 3 – Understanding CSP for Web Security

- This task explains how Content Security Policy (CSP) can be used to prevent common vulnerabilities and its appropriate use cases in web applications.
- Users can assess themselves by answering the following questions finally.

Task 3 ○ Understanding CSP for Web Security

#### Which Vulnerabilities Can CSP Prevent?

One main challenge CSP can prevent is that attackers may attempt to access your resources over an unsecure protocol.

You can use CSP to enforce the HTTPS protocol to any value defined in \*-src attributes, by adding the https:// prefix to any URL in your whitelist. This way resources will never load over an unencrypted HTTP connection. You can achieve the same effect by adding the block-all-mixed-content property.

In addition, CSP can prevent the following common vulnerabilities:

- Unsigned inline CSS statements in <style> tags
- Unsigned inline Javascript in <script> tags
- Dynamic CSS using CSSStyleSheet.insertRule()
- Dynamic Javascript code using eval()

It is best to keep the script and CSS in separate files that are referenced by the HTML page. If your site needs to allow this, you can enable it using the keywords unsafe-eval and unsafe-inline.

#### When to Use CSP

In general, complex web applications are more sensitive to XSS, making CSP important to use.

Use CSP for any application that manages sensitive data, such as administrative user interfaces, device management consoles, or any products hosting files, documents or messages created by users. In modern frameworks, adding CSP is easy and can provide high return of investment in terms of added security.

#### When Not to Use CSP

CSP may not be the best choice in these cases:

Static applications hosted on their own domains or subdomains without login or cookies.

Applications that have experienced XSS in the past or have known vulnerabilities in templates or frameworks they are using. In this case the best approach is to invest in patching or fixing vulnerable code, because CSP on its own will not provide sufficient protection. CSP should be added on top of a secure application with no known vulnerabilities.

#### Implementing Content Security Policy

The best way to add CSP retroactively to an entire website is to define a completely empty whitelist, essentially blocking everything. Initially, run CSP in report-only mode, which means the browser evaluates rules but does not block the content yet.

You can then review errors and see which of them should be added to the list (allowed) or not (disallowed).

The difficult part is deciding how much to block. For example, if you are using a script hosted through a CDN and you allow the addresses, you accept all traffic coming from those CDNs, which could include malicious traffic.

Running CSP in report mode for a few weeks, or at the most a few months, should give you all the possible cases of errors. When you feel your set of rules captures all relevant use cases, disable report-only and start blocking resources that are not on the whitelist.

Answer the questions below

What protocol can CSP enforce to ensure resources are loaded over a secure connection?

Submit

Hint



Which of the following vulnerabilities can **CSP** prevent?

- A) Unsigned inline CSS
- B) SQL Injection
- C) Buffer Overflow
- D) Cross-Site Request Forgery (CSRF)

Submit

Hint

When should CSP NOT be used?

Submit

Hint

What CSP keyword allows **inline scripts** and **dynamic code execution**?

Submit

Hint

What is the purpose of running CSP in **report-only mode**?

Submit

Answer the questions below

What protocol can CSP enforce to ensure resources are loaded over a secure connection?

CSP can enforce the HTTPS protocol by adding the https:// prefix to URLs in the whitelist.

✓ Correct Answer

Hint

Which of the following vulnerabilities can **CSP** prevent?

- A) Unsigned inline CSS
- B) SQL Injection
- C) Buffer Overflow
- D) Cross-Site Request Forgery (CSRF)

A) Unsigned inline CSS

✓ Correct Answer

Hint

When should CSP NOT be used?

CSP should not be used for static applications without login or cookies or for applications with known vulnerabilities.

✓ Correct Answer

Hint

What CSP keyword allows **inline scripts** and **dynamic code execution**?

The unsafe-inline and unsafe-eval keywords

✓ Correct Answer

Hint

What is the purpose of running CSP in **report-only mode**?

To evaluate errors without blocking content and refine the whitelist.

✓ Correct Answer

## 4. Task 4 - Understanding the Limitations of Content Security Policy (CSP)

- This section elaborates on learning the practical challenges and limitations involved in creating, managing and relying on CSP for securing web applications.
- By answering the following questions users can clarify the content well.

Task 4 Understanding the Limitations of Content Security Policy (CSP) ^

### What Are the Limitations of a Content Security Policy?

While CSP headers are a powerful security tool, they are not without their limitations:

1. Creating CSPs can be a challenging task. Before establishing a Content-Security-Policy (CSP), you must identify which domains and subdomains can access your website and what resources they may load, such as JavaScript, CSS, or PHP. This data can be laborious to gather and inventory without dedicated tools, taking expensive time and effort that can be better spent on other tasks. Once you have this information, you can begin crafting your CSP, which may span several days or weeks. Assuming that you have compiled a comprehensive list of the scripts, resources, and integrations running on the client-side, the task of understanding the functionality of each presents another unique challenge. Additionally, there is always the risk of misconfiguration. If not correctly set, CSP headers can inadvertently block legitimate content or allow the execution of malicious scripts.
2. Managing CSPs is a demanding undertaking. After constructing and implementing your CSP, someone must continuously update it with each website release. Additionally, they must monitor the browser console log for any reported CSP violations. This maintenance process is labor-intensive and necessitates a monitoring solution capable of notifying you when the CSP blocks content. Furthermore, CSP alone does not provide meaningful and actionable insights crucial for your security posture. For example, what if a resource or script that has been previously reviewed and approved is now compromised and exfiltrating data to a malicious actor?
3. CSPs are notably intricate. Most CSPs have numerous configuration lines, and the documentation can appear overwhelming, even for those with technical expertise. Specifying the resource types authorized for loading by various domains and subdomains is a complex task that offers no room for error.

Creating and managing a Content Security Policy is time-consuming and complex. However, inventorying, aggregating, and understanding what each resource is doing after it has been discovered complicates things even further. CSP headers alone do

Answer the questions below

Why can creating a Content Security Policy (CSP) be a time-consuming task?

-----, -----, -----

Which of the following is a limitation of CSP management?

A) It requires no monitoring after deployment.  
B) It needs continuous updates and violation monitoring.  
C) It blocks malicious scripts automatically without any review.  
D) It can detect when an approved script becomes malicious.

-----

Can a Content Security Policy (CSP) alone fully prevent all client-side attacks?

A) Yes, CSP is enough to stop all client-side attacks.  
B) No, CSP must be combined with other security measures.  
C) Yes, if the CSP is correctly configured.  
D) CSP can only prevent SQL Injection attacks.

-----.



Answer the questions below

Why can creating a Content Security Policy (CSP) be a time-consuming task?

Because it requires identifying all domains, subdomains, and resources used by the website and understanding their functionality

✓ Correct Answer

💡 Hint

Which of the following is a limitation of CSP management?

- A) It requires no monitoring after deployment.
- B) It needs continuous updates and violation monitoring.
- C) It blocks malicious scripts automatically without any review.
- D) It can detect when an approved script becomes malicious.

B) It needs continuous updates and violation monitoring

✓ Correct Answer

💡 Hint

Can a Content Security Policy (CSP) alone fully prevent all client-side attacks?

- A) Yes, CSP is enough to stop all client-side attacks.
- B) No, CSP must be combined with other security measures.
- C) Yes, if the CSP is correctly configured.
- D) CSP can only prevent SQL Injection attacks.

B) No, CSP must be combined with other security measures.

✓ Correct Answer

💡 Hint

## 5. Task 5 - Hand on CSP digging

- This task gives users gain some hand on experience related to Content Security Policy Misconfigurations identification.

Task 5 Hand on CSP digging

Your goal is to analyze the HTTP response headers for the provided vulnerable web application.

Start Machine

Open the website in your browser.

Open Developer Tools (F12) > Network Tab > Refresh Page > Click on the page request > Look at the 'Response Headers'.

Find the \*\*Content-Security-Policy\*\* header.

Answer the questions below

What is the CSP header defined in the response?

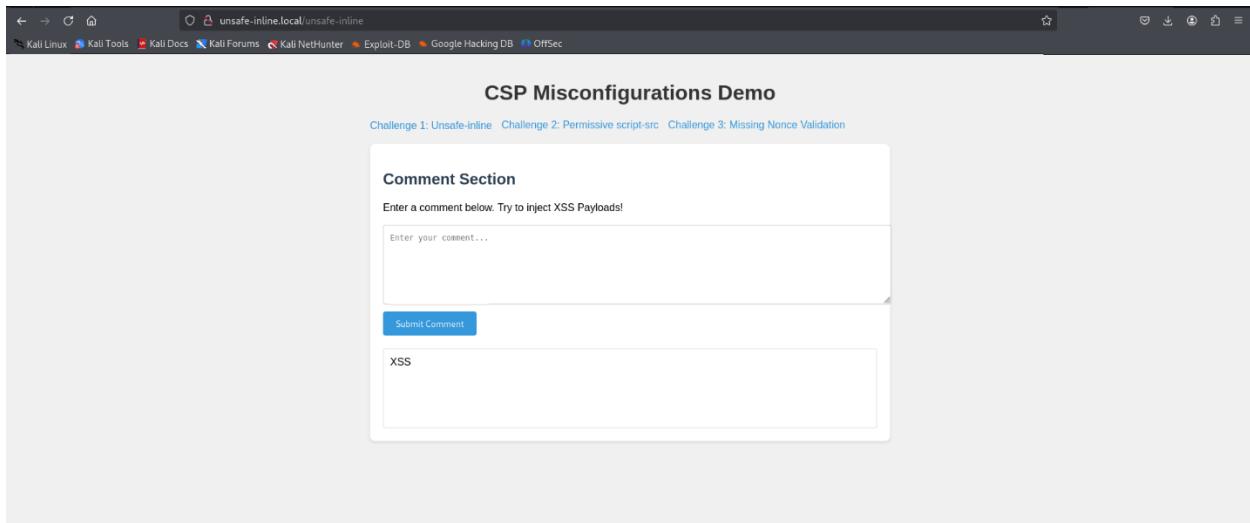
Answer format: \*\*\*\*\* Submit

What text is shown in the alert box?

Answer format: \*\*\*\*\* Submit

- As instructed on the task go to the developer tools and look for the response headers, identify the CSP header defined there.

The screenshot shows a web browser window with the URL `unsafe-inline.local/unsafe-inline`. The title bar reads "CSP Misconfigurations Demo". Below the title, there are three challenges listed: "Challenge 1: Unsafe-inline", "Challenge 2: Permissive script-src", and "Challenge 3: MissingNonce Validation". The main content area contains a "Comment Section" with the following text:  
Comment Section  
Enter a comment below. Try to inject XSS Payloads!  
Enter your comment...  
Submit Comment



Headers Cookies Request Response Timings

Filter Headers Block Resend

Transferred	2.32 kB (2.06 kB size)
Request Priority	Highest
DNS Resolution	System

Response Headers (256 B) Raw

Connection: close
Content-Length: 2064
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline'
Content-Type: text/html; charset=utf-8
Date: Wed, 30 Apr 2025 05:44:21 GMT
Server: Werkzeug/3.1.3 Python/3.10.12

Request Headers (379 B) Raw

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
---

Answer the questions below

What is the CSP header defined in the response?

"default-src 'self'; script-src 'self' 'unsafe-inline'"

✓ Correct Answer

What text is shown in the alert box?

XSS Bypass!

✓ Correct Answer

## 6. Task 6 - CSP Bypass — Inline Event Handler Challenge

- This task shows though it is protected by a Content Security Policy that restricts JavaScript execution, how it is still vulnerable due to a weak policy allowing 'unsafe-inline'.
- Task Instructions:
  1. Download and open the HTML file in your browser.
  2. You will see an input field and a button labeled **Run Payload**.
  3. Inject a payload that abuses the CSP misconfiguration.
  4. Your goal is to make an alert box appear or retrieve the hidden flag.

Task 6 ○ CSP Bypass — Inline Event Handler Challenge

Download and open the HTML file provided with this task. This file is protected with a Content Security Policy that restricts JavaScript execution — however, due to a weak policy allowing "unsafe-inline", it is still vulnerable.

When you run your payload successfully, it will trigger an alert box or reveal a flag.



### Task Instructions:

1. Download and open the HTML file in your browser.
2. You will see an input field and a button labeled **Run Payload**.
3. Inject a payload that abuses the CSP misconfiguration.
4. Your goal is to make an alert box appear or retrieve the hidden flag.

### Answer the questions below

What CSP directive is weakening the policy and allowing XSS?

- a. 'self'
  - b. 'unsafe-inline'
  - c. script-src
  - d. 'strict-dynamic'

Answer format: \*.\*\*\*\*\*

 Submit

Hint

Which payload successfully triggers an alert box?

- a. <script>alert(1)</script>
  - b. <img src=x onerror=alert(1)>
  - c. <svg onload=alert(1)>
  - d. all of the above

 Submit

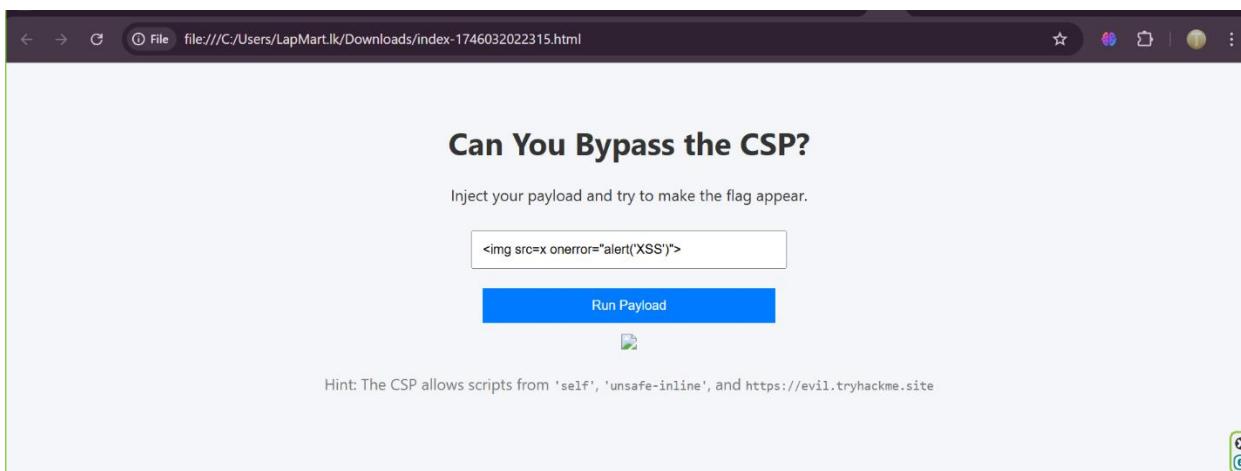
Hint

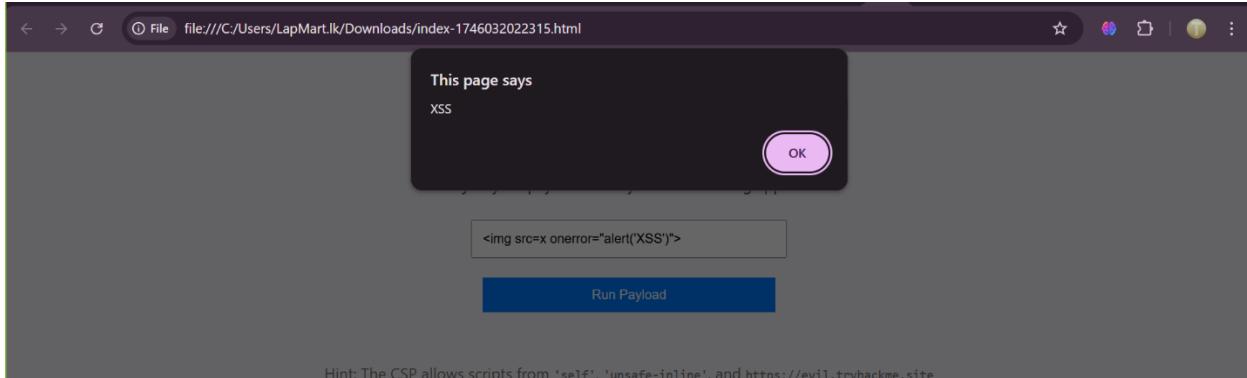
What would be a more secure CSP setting to prevent this XSS?

- a. Remove 'self'
  - b. Add 'unsafe-eval'
  - c. Remove 'unsafe-inline' from script-src
  - d. Allow all domains

 Submit

- When you download the file, you can open it through a browser and visit this simple page. Then give a payload to identify the inline CSP vulnerability.





Answer the questions below

What CSP directive is weakening the policy and allowing XSS?

- a. 'self'
- b. 'unsafe-inline'
- c. script-src
- d. 'strict-dynamic'

b. 'unsafe-inline'

✓ Woop woop! Your answer is correct

✓ Correct Answer

✗ Hint

Which payload successfully triggers an alert box?

- a. <script>alert(1)</script>
- b. <img src=x onerror=alert(1)>
- c. <svg onload=alert(1)>
- d. all of the above

b. <img src=x onerror=alert(1)>

✓ Correct Answer

✗ Hint

What would be a more secure CSP setting to prevent this XSS?

- a. Remove 'self'
- b. Add 'unsafe-eval'
- c. Remove 'unsafe-inline' from script-src
- d. Allow all domains

c. Remove 'unsafe-inline' from script-src

✓ Correct Answer

## 7. Task 7 – [Vulnerable Web App]

- Welcome to the Vulnerable Web App designed to simulate common web vulnerabilities in a safe, sandboxed environment. In this task, you'll explore and exploit three common flaws found in insecure web applications. Each challenge demonstrates a different vulnerability and includes hints to guide your testing process.

The screenshot shows a web-based application interface for 'Task 7 [Vulnerable Web App]'. At the top, a green bar indicates 'Room completed (100%)'. Below it, the title 'Task 7 [Vulnerable Web App]' is shown with a checkmark. On the right, there are download and refresh icons. The main content area lists three tasks:

- Task 1: Sensitive Data Exposure**
  - Inspect the page source for hidden comments left by developers. These comments may contain sensitive information such as passwords or credentials that shouldn't be publicly accessible.
- Task 2: HTML Injection**
  - Test the input field that displays user-submitted content. The application fails to sanitize the input, allowing you to inject HTML code into the page. Try inserting a clickable malicious link to simulate a phishing attack.
    - ✓ First give your name, it will be displayed on the page
    - ✓ Next under Hint a HTML tag is their, try injecting it and observe the output
- Task 3: File Upload Vulnerability**

Upload a file to the server through an unprotected file upload form. The system does not validate file types correctly, making it possible to upload potentially dangerous files (e.g., a PHP shell disguised as an image). The app detects .php files but simulates a real-world vulnerable scenario.

On the right side of the interface, there is a blue button labeled 'Download Task Files'.

Answer the questions below

What is the password found in the HTML source comment?

tryhackme123

✓ Correct Answer

What is the flag revealed after successfully injecting a malicious link?

HTML\_INJ3CTION

✓ Correct Answer

What kind of file can you upload to exploit the file upload vulnerability?

shell.php.jpg

✓ Correct Answer

💡 Hint

What flag is displayed after successfully uploading a potentially malicious file?

FILE\_UPLOAD\_VULN

✓ Correct Answer

```

30     }
31     #output {
32       margin-top: 10px;
33       padding: 10px;
34       border: 1px solid #ddd;
35   }
36   .hint {
37     color: #555;
38     font-style: italic;
39   }
40   <!-- Hidden comment with a clue for users -->
41   <!-- Hint: Check the page source for sensitive information. Password: tryhackme123 -->
42 </style>
43 </head>
44 <body>
45   <div class="container">
46     <h1>Vulnerable Web App - TryHackMe Exploitation Task</h1>
47
48     <div class="section">
49       <h2>Task 1: Sensitive Data Exposure</h2>
50       <p>Inspect the page source code to find hidden information, such as a password. Right-click and select "View Page Source" to start.</p>
51       <p class="hint">Hint: Developers sometimes leave sensitive data in comments.</p>
52     </div>
53
54     <div class="section">
55       <h2>Task 2: HTML Injection</h2>
56       <p>Enter your name below. The input is displayed on the page, but it's not sanitized. Can you inject HTML to create a malicious link?</p>
57       <form id="nameForm">
58         <label for="nameInput">Your Name:</label><br>
59         <input type="text" id="nameInput" placeholder="Enter your name">
60         <button type="submit">Submit</button>
61       </form>
62       <div id="output"></div>
63       <p class="hint">Hint: Try injecting HTML like <a href="http://hacker.com">Click Me</a>.</p>
64     </div>
65
66     <div class="section">
67       ...
68     </div>
69   </div>
70
71   <div class="section">
72     ...
73   </div>
74
75   <div class="section">
76     ...
77   </div>
78
79   <div class="section">
80     ...
81   </div>
82
83   <div class="section">
84     ...
85   </div>
86
87   <div class="section">
88     ...
89   </div>
90
91   <div class="section">
92     ...
93   </div>
94
95   <div class="section">
96     ...
97   </div>
98
99   <div class="section">
100    ...
101  </div>
102
103  <div class="section">
104    ...
105  </div>
106
107  <div class="section">
108    ...
109  </div>
110
111  <div class="section">
112    ...
113  </div>
114
115  <div class="section">
116    ...
117  </div>
118
119  <div class="section">
120    ...
121  </div>
122
123  <div class="section">
124    ...
125  </div>
126
127  <div class="section">
128    ...
129  </div>
130
131  <div class="section">
132    ...
133  </div>
134
135  <div class="section">
136    ...
137  </div>
138
139  <div class="section">
140    ...
141  </div>
142
143  <div class="section">
144    ...
145  </div>
146
147  <div class="section">
148    ...
149  </div>
150
151  <div class="section">
152    ...
153  </div>
154
155  <div class="section">
156    ...
157  </div>
158
159  <div class="section">
160    ...
161  </div>
162
163  <div class="section">
164    ...
165  </div>
166
167  <div class="section">
168    ...
169  </div>
170
171  <div class="section">
172    ...
173  </div>
174
175  <div class="section">
176    ...
177  </div>
178
179  <div class="section">
180    ...
181  </div>
182
183  <div class="section">
184    ...
185  </div>
186
187  <div class="section">
188    ...
189  </div>
190
191  <div class="section">
192    ...
193  </div>
194
195  <div class="section">
196    ...
197  </div>
198
199  <div class="section">
200    ...
201  </div>
202
203  <div class="section">
204    ...
205  </div>
206
207  <div class="section">
208    ...
209  </div>
210
211  <div class="section">
212    ...
213  </div>
214
215  <div class="section">
216    ...
217  </div>
218
219  <div class="section">
220    ...
221  </div>
222
223  <div class="section">
224    ...
225  </div>
226
227  <div class="section">
228    ...
229  </div>
230
231  <div class="section">
232    ...
233  </div>
234
235  <div class="section">
236    ...
237  </div>
238
239  <div class="section">
240    ...
241  </div>
242
243  <div class="section">
244    ...
245  </div>
246
247  <div class="section">
248    ...
249  </div>
250
251  <div class="section">
252    ...
253  </div>
254
255  <div class="section">
256    ...
257  </div>
258
259  <div class="section">
260    ...
261  </div>
262
263  <div class="section">
264    ...
265  </div>
266
267  <div class="section">
268    ...
269  </div>
270
271  <div class="section">
272    ...
273  </div>
274
275  <div class="section">
276    ...
277  </div>
278
279  <div class="section">
280    ...
281  </div>
282
283  <div class="section">
284    ...
285  </div>
286
287  <div class="section">
288    ...
289  </div>
290
291  <div class="section">
292    ...
293  </div>
294
295  <div class="section">
296    ...
297  </div>
298
299  <div class="section">
300    ...
301  </div>
302
303  <div class="section">
304    ...
305  </div>
306
307  <div class="section">
308    ...
309  </div>
310
311  <div class="section">
312    ...
313  </div>
314
315  <div class="section">
316    ...
317  </div>
318
319  <div class="section">
320    ...
321  </div>
322
323  <div class="section">
324    ...
325  </div>
326
327  <div class="section">
328    ...
329  </div>
330
331  <div class="section">
332    ...
333  </div>
334
335  <div class="section">
336    ...
337  </div>
338
339  <div class="section">
340    ...
341  </div>
342
343  <div class="section">
344    ...
345  </div>
346
347  <div class="section">
348    ...
349  </div>
350
351  <div class="section">
352    ...
353  </div>
354
355  <div class="section">
356    ...
357  </div>
358
359  <div class="section">
360    ...
361  </div>
362
363  <div class="section">
364    ...
365  </div>
366
367  <div class="section">
368    ...
369  </div>
370
371  <div class="section">
372    ...
373  </div>
374
375  <div class="section">
376    ...
377  </div>
378
379  <div class="section">
380    ...
381  </div>
382
383  <div class="section">
384    ...
385  </div>
386
387  <div class="section">
388    ...
389  </div>
390
391  <div class="section">
392    ...
393  </div>
394
395  <div class="section">
396    ...
397  </div>
398
399  <div class="section">
400    ...
401  </div>
402
403  <div class="section">
404    ...
405  </div>
406
407  <div class="section">
408    ...
409  </div>
410
411  <div class="section">
412    ...
413  </div>
414
415  <div class="section">
416    ...
417  </div>
418
419  <div class="section">
420    ...
421  </div>
422
423  <div class="section">
424    ...
425  </div>
426
427  <div class="section">
428    ...
429  </div>
430
431  <div class="section">
432    ...
433  </div>
434
435  <div class="section">
436    ...
437  </div>
438
439  <div class="section">
440    ...
441  </div>
442
443  <div class="section">
444    ...
445  </div>
446
447  <div class="section">
448    ...
449  </div>
450
451  <div class="section">
452    ...
453  </div>
454
455  <div class="section">
456    ...
457  </div>
458
459  <div class="section">
460    ...
461  </div>
462
463  <div class="section">
464    ...
465  </div>
466
467  <div class="section">
468    ...
469  </div>
470
471  <div class="section">
472    ...
473  </div>
474
475  <div class="section">
476    ...
477  </div>
478
479  <div class="section">
480    ...
481  </div>
482
483  <div class="section">
484    ...
485  </div>
486
487  <div class="section">
488    ...
489  </div>
490
491  <div class="section">
492    ...
493  </div>
494
495  <div class="section">
496    ...
497  </div>
498
499  <div class="section">
500    ...
501  </div>
502
503  <div class="section">
504    ...
505  </div>
506
507  <div class="section">
508    ...
509  </div>
510
511  <div class="section">
512    ...
513  </div>
514
515  <div class="section">
516    ...
517  </div>
518
519  <div class="section">
520    ...
521  </div>
522
523  <div class="section">
524    ...
525  </div>
526
527  <div class="section">
528    ...
529  </div>
530
531  <div class="section">
532    ...
533  </div>
534
535  <div class="section">
536    ...
537  </div>
538
539  <div class="section">
540    ...
541  </div>
542
543  <div class="section">
544    ...
545  </div>
546
547  <div class="section">
548    ...
549  </div>
550
551  <div class="section">
552    ...
553  </div>
554
555  <div class="section">
556    ...
557  </div>
558
559  <div class="section">
560    ...
561  </div>
562
563  <div class="section">
564    ...
565  </div>
566
567  <div class="section">
568    ...
569  </div>
570
571  <div class="section">
572    ...
573  </div>
574
575  <div class="section">
576    ...
577  </div>
578
579  <div class="section">
580    ...
581  </div>
582
583  <div class="section">
584    ...
585  </div>
586
587  <div class="section">
588    ...
589  </div>
590
591  <div class="section">
592    ...
593  </div>
594
595  <div class="section">
596    ...
597  </div>
598
599  <div class="section">
600    ...
601  </div>
602
603  <div class="section">
604    ...
605  </div>
606
607  <div class="section">
608    ...
609  </div>
610
611  <div class="section">
612    ...
613  </div>
614
615  <div class="section">
616    ...
617  </div>
618
619  <div class="section">
620    ...
621  </div>
622
623  <div class="section">
624    ...
625  </div>
626
627  <div class="section">
628    ...
629  </div>
630
631  <div class="section">
632    ...
633  </div>
634
635  <div class="section">
636    ...
637  </div>
638
639  <div class="section">
640    ...
641  </div>
642
643  <div class="section">
644    ...
645  </div>
646
647  <div class="section">
648    ...
649  </div>
650
651  <div class="section">
652    ...
653  </div>
654
655  <div class="section">
656    ...
657  </div>
658
659  <div class="section">
660    ...
661  </div>
662
663  <div class="section">
664    ...
665  </div>
666
667  <div class="section">
668    ...
669  </div>
670
671  <div class="section">
672    ...
673  </div>
674
675  <div class="section">
676    ...
677  </div>
678
679  <div class="section">
680    ...
681  </div>
682
683  <div class="section">
684    ...
685  </div>
686
687  <div class="section">
688    ...
689  </div>
690
691  <div class="section">
692    ...
693  </div>
694
695  <div class="section">
696    ...
697  </div>
698
699  <div class="section">
700    ...
701  </div>
702
703  <div class="section">
704    ...
705  </div>
706
707  <div class="section">
708    ...
709  </div>
710
711  <div class="section">
712    ...
713  </div>
714
715  <div class="section">
716    ...
717  </div>
718
719  <div class="section">
720    ...
721  </div>
722
723  <div class="section">
724    ...
725  </div>
726
727  <div class="section">
728    ...
729  </div>
730
731  <div class="section">
732    ...
733  </div>
734
735  <div class="section">
736    ...
737  </div>
738
739  <div class="section">
740    ...
741  </div>
742
743  <div class="section">
744    ...
745  </div>
746
747  <div class="section">
748    ...
749  </div>
750
751  <div class="section">
752    ...
753  </div>
754
755  <div class="section">
756    ...
757  </div>
758
759  <div class="section">
760    ...
761  </div>
762
763  <div class="section">
764    ...
765  </div>
766
767  <div class="section">
768    ...
769  </div>
770
771  <div class="section">
772    ...
773  </div>
774
775  <div class="section">
776    ...
777  </div>
778
779  <div class="section">
780    ...
781  </div>
782
783  <div class="section">
784    ...
785  </div>
786
787  <div class="section">
788    ...
789  </div>
790
791  <div class="section">
792    ...
793  </div>
794
795  <div class="section">
796    ...
797  </div>
798
799  <div class="section">
800    ...
801  </div>
802
803  <div class="section">
804    ...
805  </div>
806
807  <div class="section">
808    ...
809  </div>
810
811  <div class="section">
812    ...
813  </div>
814
815  <div class="section">
816    ...
817  </div>
818
819  <div class="section">
820    ...
821  </div>
822
823  <div class="section">
824    ...
825  </div>
826
827  <div class="section">
828    ...
829  </div>
830
831  <div class="section">
832    ...
833  </div>
834
835  <div class="section">
836    ...
837  </div>
838
839  <div class="section">
840    ...
841  </div>
842
843  <div class="section">
844    ...
845  </div>
846
847  <div class="section">
848    ...
849  </div>
850
851  <div class="section">
852    ...
853  </div>
854
855  <div class="section">
856    ...
857  </div>
858
859  <div class="section">
860    ...
861  </div>
862
863  <div class="section">
864    ...
865  </div>
866
867  <div class="section">
868    ...
869  </div>
870
871  <div class="section">
872    ...
873  </div>
874
875  <div class="section">
876    ...
877  </div>
878
879  <div class="section">
880    ...
881  </div>
882
883  <div class="section">
884    ...
885  </div>
886
887  <div class="section">
888    ...
889  </div>
890
891  <div class="section">
892    ...
893  </div>
894
895  <div class="section">
896    ...
897  </div>
898
899  <div class="section">
900    ...
901  </div>
902
903  <div class="section">
904    ...
905  </div>
906
907  <div class="section">
908    ...
909  </div>
910
911  <div class="section">
912    ...
913  </div>
914
915  <div class="section">
916    ...
917  </div>
918
919  <div class="section">
920    ...
921  </div>
922
923  <div class="section">
924    ...
925  </div>
926
927  <div class="section">
928    ...
929  </div>
930
931  <div class="section">
932    ...
933  </div>
934
935  <div class="section">
936    ...
937  </div>
938
939  <div class="section">
940    ...
941  </div>
942
943  <div class="section">
944    ...
945  </div>
946
947  <div class="section">
948    ...
949  </div>
950
951  <div class="section">
952    ...
953  </div>
954
955  <div class="section">
956    ...
957  </div>
958
959  <div class="section">
960    ...
961  </div>
962
963  <div class="section">
964    ...
965  </div>
966
967  <div class="section">
968    ...
969  </div>
970
971  <div class="section">
972    ...
973  </div>
974
975  <div class="section">
976    ...
977  </div>
978
979  <div class="section">
980    ...
981  </div>
982
983  <div class="section">
984    ...
985  </div>
986
987  <div class="section">
988    ...
989  </div>
990
991  <div class="section">
992    ...
993  </div>
994
995  <div class="section">
996    ...
997  </div>
998
999  <div class="section">
1000   ...
1001  </div>
1002
1003  <div class="section">
1004   ...
1005  </div>
1006
1007  <div class="section">
1008   ...
1009  </div>
1010
1011  <div class="section">
1012   ...
1013  </div>
1014
1015  <div class="section">
1016   ...
1017  </div>
1018
1019  <div class="section">
1020   ...
1021  </div>
1022
1023  <div class="section">
1024   ...
1025  </div>
1026
1027  <div class="section">
1028   ...
1029  </div>
1030
1031  <div class="section">
1032   ...
1033  </div>
1034
1035  <div class="section">
1036   ...
1037  </div>
1038
1039  <div class="section">
1040   ...
1041  </div>
1042
1043  <div class="section">
1044   ...
1045  </div>
1046
1047  <div class="section">
1048   ...
1049  </div>
1050
1051  <div class="section">
1052   ...
1053  </div>
1054
1055  <div class="section">
1056   ...
1057  </div>
1058
1059  <div class="section">
1060   ...
1061  </div>
1062
1063  <div class="section">
1064   ...
1065  </div>
1066
1067  <div class="section">
1068   ...
1069  </div>
1070
1071  <div class="section">
1072   ...
1073  </div>
1074
1075  <div class="section">
1076   ...
1077  </div>
1078
1079  <div class="section">
1080   ...
1081  </div>
1082
1083  <div class="section">
1084   ...
1085  </div>
1086
1087  <div class="section">
1088   ...
1089  </div>
1090
1091  <div class="section">
1092   ...
1093  </div>
1094
1095  <div class="section">
1096   ...
1097  </div>
1098
1099  <div class="section">
1100   ...
1101  </div>
1102
1103  <div class="section">
1104   ...
1105  </div>
1106
1107  <div class="section">
1108   ...
1109  </div>
1110
1111  <div class="section">
1112   ...
1113  </div>
1114
1115  <div class="section">
1116   ...
1117  </div>
1118
1119  <div class="section">
1120   ...
1121  </div>
1122
1123  <div class="section">
1124   ...
1125  </div>
1126
1127  <div class="section">
1128   ...
1129  </div>
1130
1131  <div class="section">
1132   ...
1133  </div>
1134
1135  <div class="section">
1136   ...
1137  </div>
1138
1139  <div class="section">
1140   ...
1141  </div>
1142
1143  <div class="section">
1144   ...
1145  </div>
1146
1147  <div class="section">
1148   ...
1149  </div>
1150
1151  <div class="section">
1152   ...
1153  </div>
1154
1155  <div class="section">
1156   ...
1157  </div>
1158
1159  <div class="section">
1160   ...
1161  </div>
1162
1163  <div class="section">
1164   ...
1165  </div>
1166
1167  <div class="section">
1168   ...
1169  </div>
1170
1171  <div class="section">
1172   ...
1173  </div>
1174
1175  <div class="section">
1176   ...
1177  </div>
1178
1179  <div class="section">
1180   ...
1181  </div>
1182
1183  <div class="section">
1184   ...
1185  </div>
1186
1187  <div class="section">
1188   ...
1189  </div>
1190
1191  <div class="section">
1192   ...
1193  </div>
1194
1195  <div class="section">
1196   ...
1197  </div>
1198
1199  <div class="section">
1200   ...
1201  </div>
1202
1203  <div class="section">
1204   ...
1205  </div>
1206
1207  <div class="section">
1208   ...
1209  </div>
1210
1211  <div class="section">
1212   ...
1213  </div>
1214
1215  <div class="section">
1216   ...
1217  </div>
1218
1219  <div class="section">
1220   ...
1221  </div>
1222
1223  <div class="section">
1224   ...
1225  </div>
1226
1227  <div class="section">
1228   ...
1229  </div>
1230
1231  <div class="section">
1232   ...
1233  </div>
1234
1235  <div class="section">
1236   ...
1237  </div>
1238
1239  <div class="section">
1240   ...
1241  </div>
1242
1243  <div class="section">
1244   ...
1245  </div>
1246
1247  <div class="section">
1248   ...
1249  </div>
1250
1251  <div class="section">
1252   ...
1253  </div>
1254
1255  <div class="section">
1256   ...
1257  </div>
1258
1259  <div class="section">
1260   ...
1261  </div>
1262
1263  <div class="section">
1264   ...
1265  </div>
1266
1267  <div class="section">
1268   ...
1269  </div>
1270
1271  <div class="section">
1272   ...
1273  </div>
1274
1275  <div class="section">
1276   ...
1277  </div>
1278
1279  <div class="section">
1280   ...
1281  </div>
1282
1283  <div class="section">
1284   ...
1285  </div>
1286
1287  <div class="section">
1288   ...
1289  </div>
1290
1291  <div class="section">
1292   ...
1293  </div>
1294
1295  <div class="section">
1296   ...
1297  </div>
1298
1299  <div class="section">
1300   ...
1301  </div>
1302
1303  <div class="section">
1304   ...
1305  </div>
1306
1307  <div class="section">
1308   ...
1309  </div>
1310
1311  <div class="section">
1312   ...
1313  </div>
1314
1315  <div class="section">
1316   ...
1317  </div>
1318
1319  <div class="section">
1320   ...
1321  </div>
1322
1323  <div class="section">
1324   ...
1325  </div>
1326
1327  <div class="section">
1328   ...
1329  </div>
1330
1331  &
```

### Task 3: File Upload Vulnerability

Upload a file to the server. The server doesn't validate file types properly. Can you upload a malicious file?

Choose a file:

shell.php.jpg

File uploaded: shell.php.jpg

Flag: FILE\_UPLOAD\_VULN - You uploaded a potentially malicious file!

*Hint: Try uploading a file with a .php extension disguised as an image (e.g., shell.php.jpg).*

## Room Link

**Room Link:** <https://tryhackme.com/jr/crackingweakcspmisconfigs>

# Reflection

## 1. Personal Insights and Lessons Learned While Designing the Room

- Understanding the Limits of CSP:

While designing the CSP challenge, I realized that many developers overestimate the protection offered by CSP. Simply applying a policy is not enough — if it includes overly permissive directives or allows scripts from compromised or untrusted domains, it can still leave the application exposed. This experience taught me how critical it is to implement strict, minimal policies.

- Creating Realistic Bypass Scenarios:

Building a lab to demonstrate CSP bypasses helped me understand the subtle ways attackers can exploit relaxed directives, such as unsafe-inline or script-src with overly broad whitelists. Crafting scenarios where a supposedly secure CSP could still be circumvented was both challenging and rewarding.

- Thinking Like an Attacker:

Developing the lab deepened my attacker mindset. I had to consider how a user might interact with the page, how a malicious actor could inject or manipulate content, and how different browser behaviors might influence exploitation. This perspective is essential not only for breaking things but also for building more secure systems.

- Technical Deployment Skills:

Setting up the environment required configuring a Flask-based web app with a vulnerable CSP policy, ensuring it served the correct headers, and testing it across browsers. I also refined my skills in deploying services on Ubuntu, using systemd for autostart and managing port binding and access within TryHackMe's VM infrastructure.

## 2. How This Room Contributes to the Learning Community

- Hands-On CSP Misconfiguration Exploration:

This room offers a practical way to explore how small missteps in CSP implementation can lead to significant vulnerabilities. It moves beyond theory and allows learners to experiment in a safe, controlled environment.

- Promoting Secure Development Practices:

By highlighting the consequences of weak CSP rules, the room reinforces the need for careful configuration. Learners gain a better appreciation for why directives should be as restrictive as possible and how misconfigurations can backfire.

- **Progressive Learning Experience:**  
The challenge is structured to guide users through discovery and exploitation without being overwhelming. This makes it accessible to beginners while still engaging for those with intermediate skills.
- **Encouraging Creative Problem Solving:**  
Users are prompted to find non-obvious paths, such as leveraging whitelisted domains or crafting specific payloads that slip through loose policies. This nurtures critical thinking and creative exploitation techniques.
- **Real-World Relevance:**  
CSP misconfigurations are a common issue in real-world bug bounty programs and pentests. By practicing in this environment, learners develop transferable skills for recognizing and responsibly reporting similar issues in production systems.

# Workings

## Create Virtual Machine

- Select the correct type and version, here I select *ubuntu-22.04.5-live-server-amd64.iso*.

The screenshot shows a list of Ubuntu 22.04.5 desktop and server ISO files. The 'ubuntu-22.04.5-live-server-amd64.iso' file is selected, displaying its details:

File	Size	Last Modified	Description
ubuntu-22.04.5-desktop-amd64.iso	4.4G	2024-09-11 14:38	Desktop image for 64-bit PC (AMD64) computers (standard download)
ubuntu-22.04.5-desktop-amd64.iso.torrent	355K	2024-09-12 18:13	Desktop image for 64-bit PC (AMD64) computers (BitTorrent download)
ubuntu-22.04.5-desktop-amd64.iso.zsync	10M	2024-09-12 18:13	Desktop image for 64-bit PC (AMD64) computers (zsync metafile)
ubuntu-22.04.5-desktop-amd64.list	26K	2024-09-11 14:39	Desktop image for 64-bit PC (AMD64) computers (file listing)
ubuntu-22.04.5-desktop-amd64.manifest	60K	2024-09-11 14:32	Desktop image for 64-bit PC (AMD64) computers (contents of live filesystem)
ubuntu-22.04.5-live-server-amd64.iso	2.0G	2024-09-11 18:46	Server install image for 64-bit PC (AMD64) computers (standard download)
ubuntu-22.04.5-live-server-amd64.iso.torrent	160K	2024-09-12 18:16	Server install image for 64-bit PC (AMD64) computers (BitTorrent download)
ubuntu-22.04.5-live-server-amd64.iso.zsync	4.0M	2024-09-12 18:16	Server install image for 64-bit PC (AMD64) computers (zsync metafile)
ubuntu-22.04.5-live-server-amd64.list	8.5K	2024-09-11 18:46	Server install image for 64-bit PC (AMD64) computers (file listing)
ubuntu-22.04.5-live-server-			Server install image for 64-bit PC (AMD64)

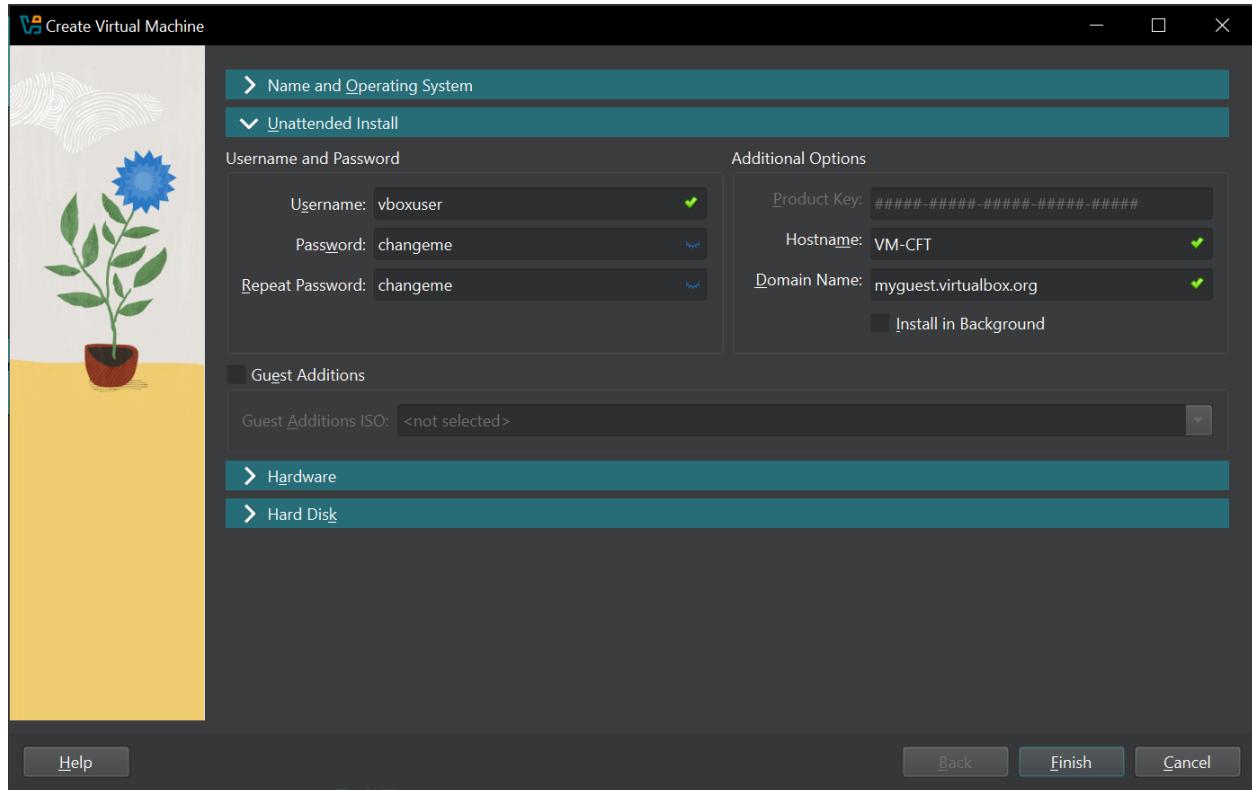
- Click 'New' button to open a dialog. Type a name for the new virtual machine.

The screenshot shows the 'Create Virtual Machine' dialog in VirtualBox. The 'Name and Operating System' tab is selected, displaying the following configuration:

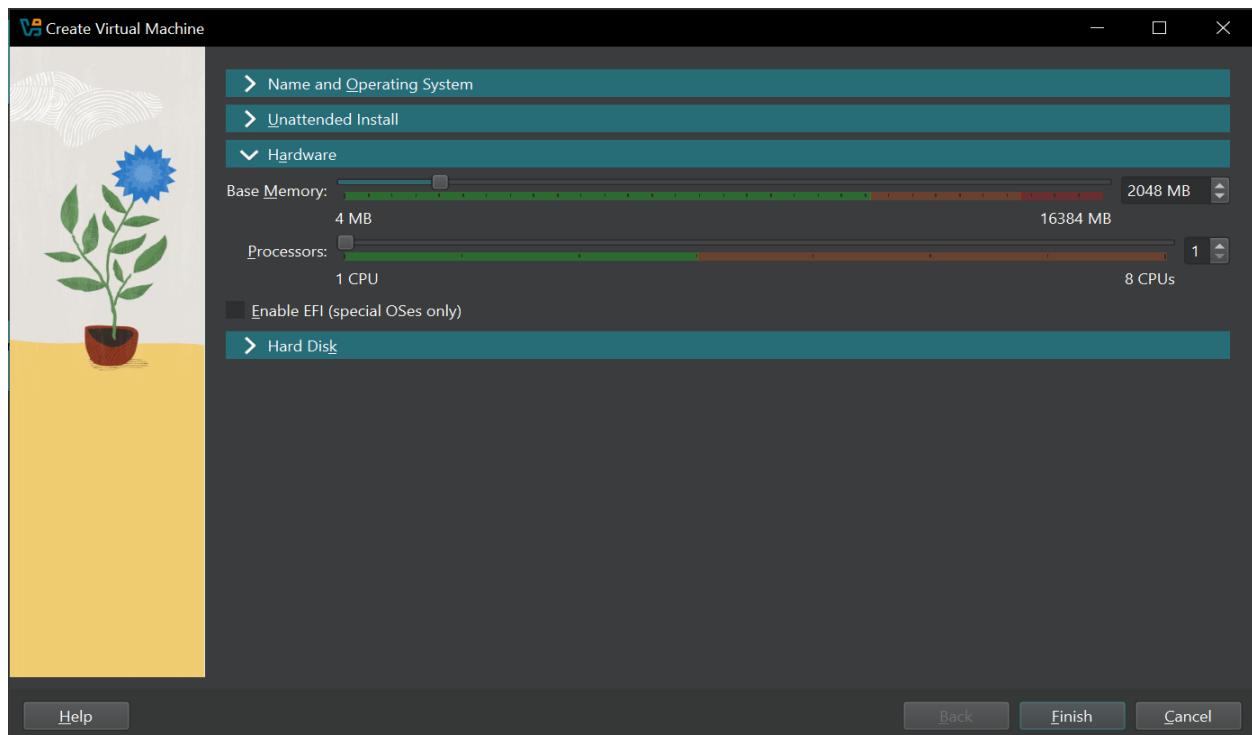
- Name: VM-CFT
- Folder: C:\Users\LapMart.Ik\VirtualBox VMs
- ISO Image: D:\App Setups\Ubuntu\ubuntu-22.04.5-live-server-amd64.iso
- Edition:
- Type: Linux
- Subtype: Ubuntu
- Version: Ubuntu (64-bit)
- Skip Unattended Installation (checkbox checked)

Below the main configuration, there are three tabs: Unattended Install, Hardware, and Hard Disk.

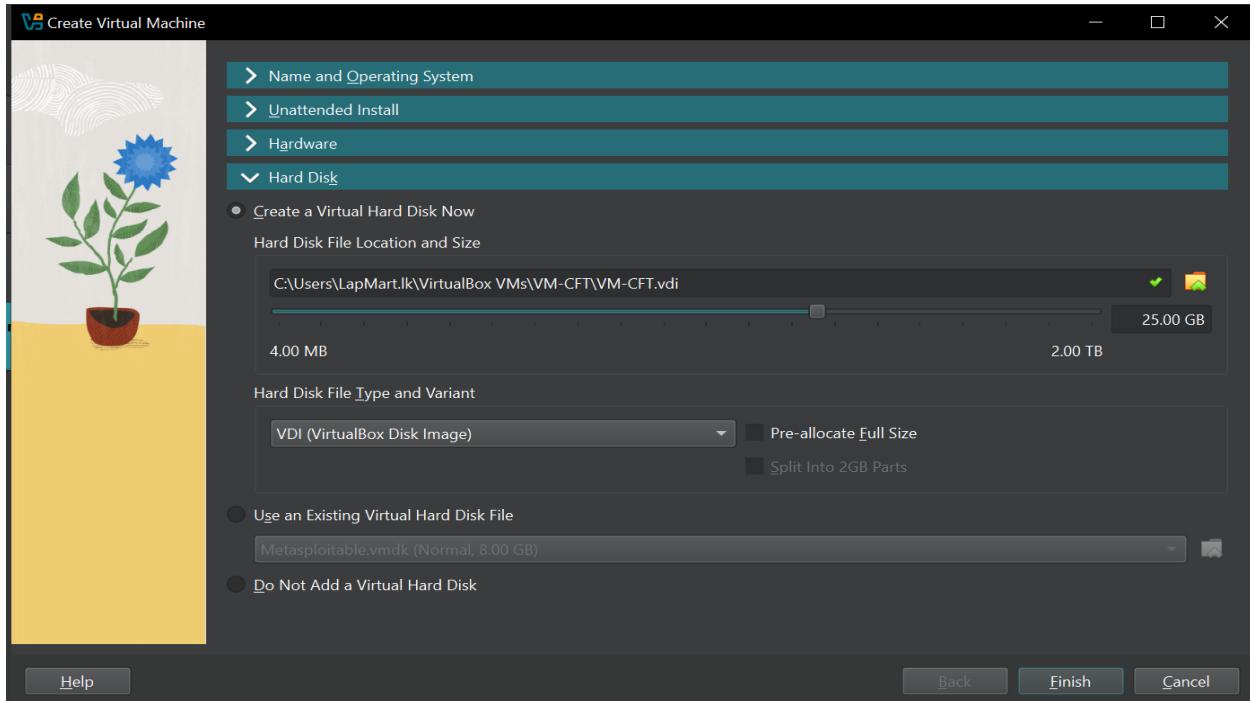
- Give username password and next



- Allocate the memory and CPU (2GB,2cpu recommended).

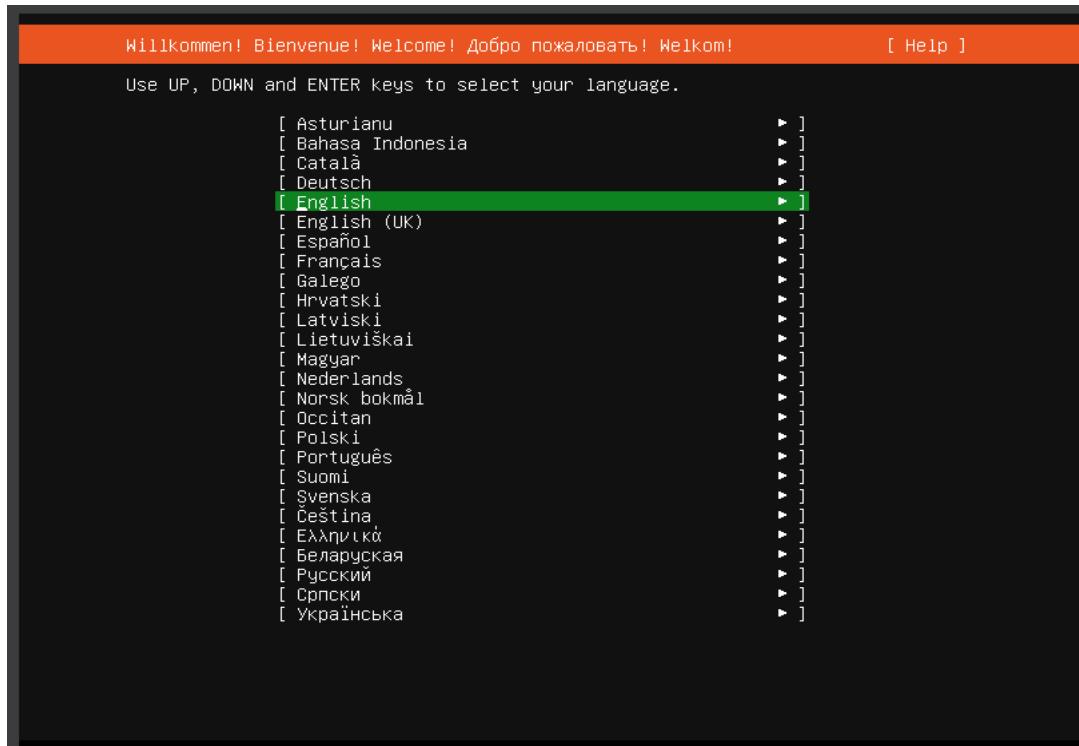


- Give Hard space as 20 GB and finish.

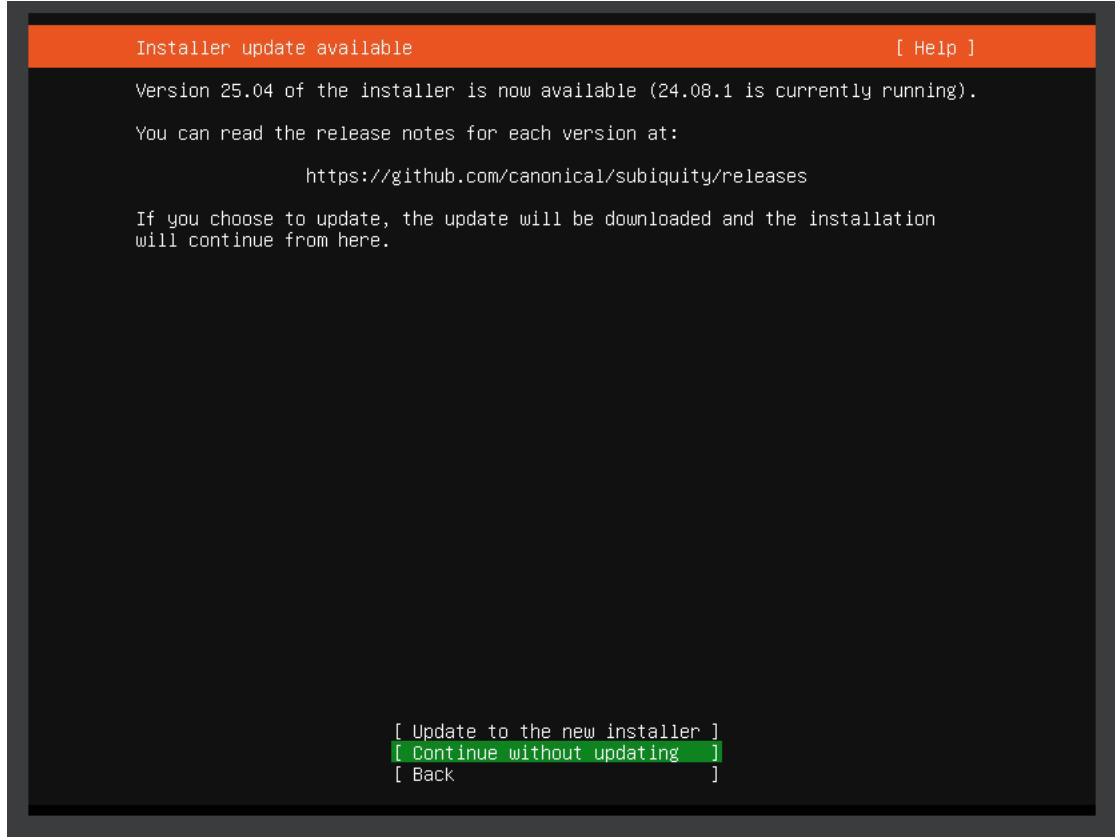


## Ubuntu Server Installation

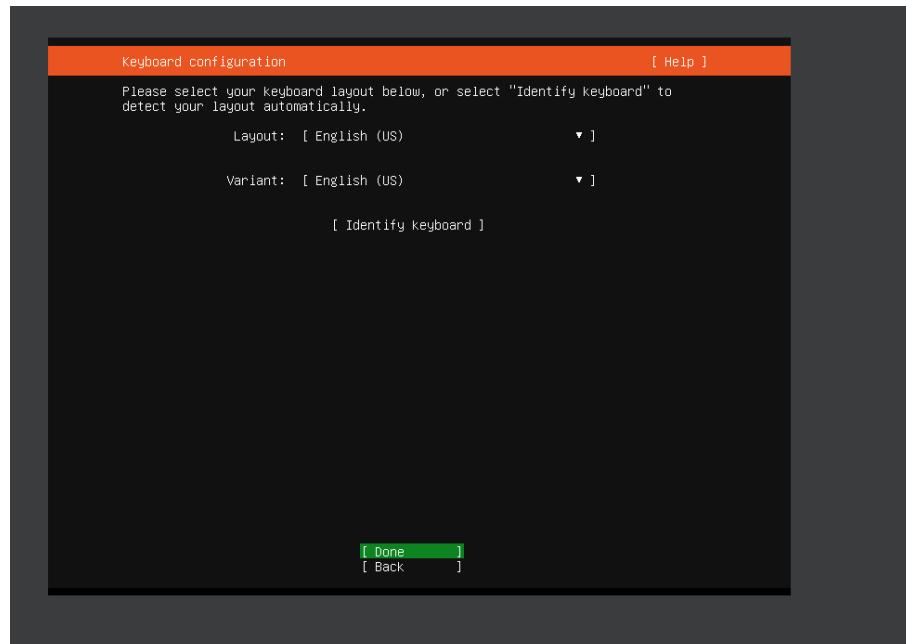
- We will first see this window where a language needs to be selected.

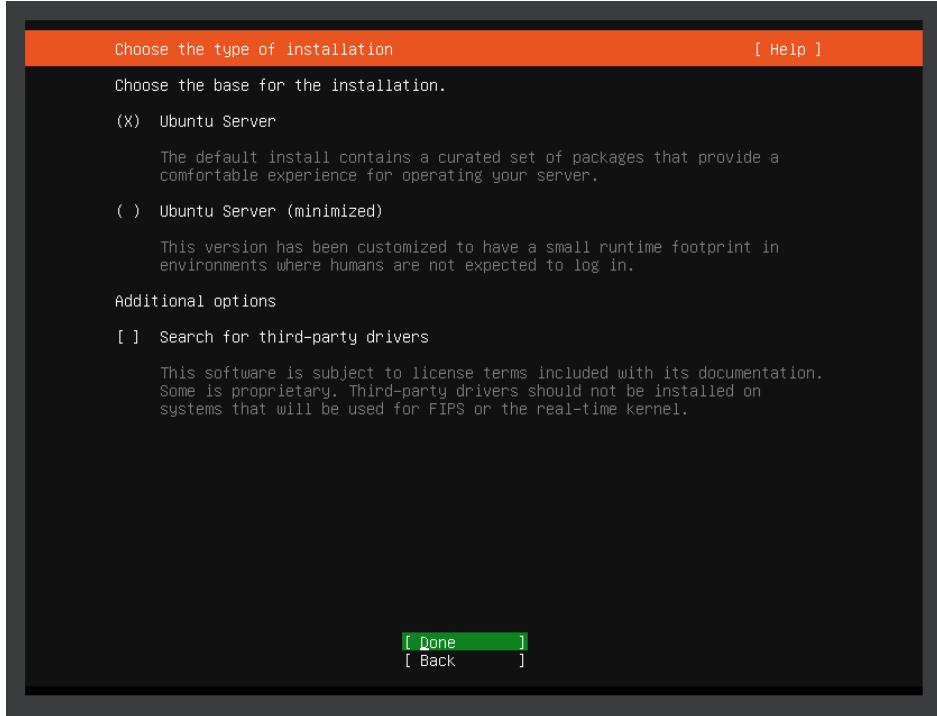


- Press Enter and we will see the installer update available. But let's continue without updating it.

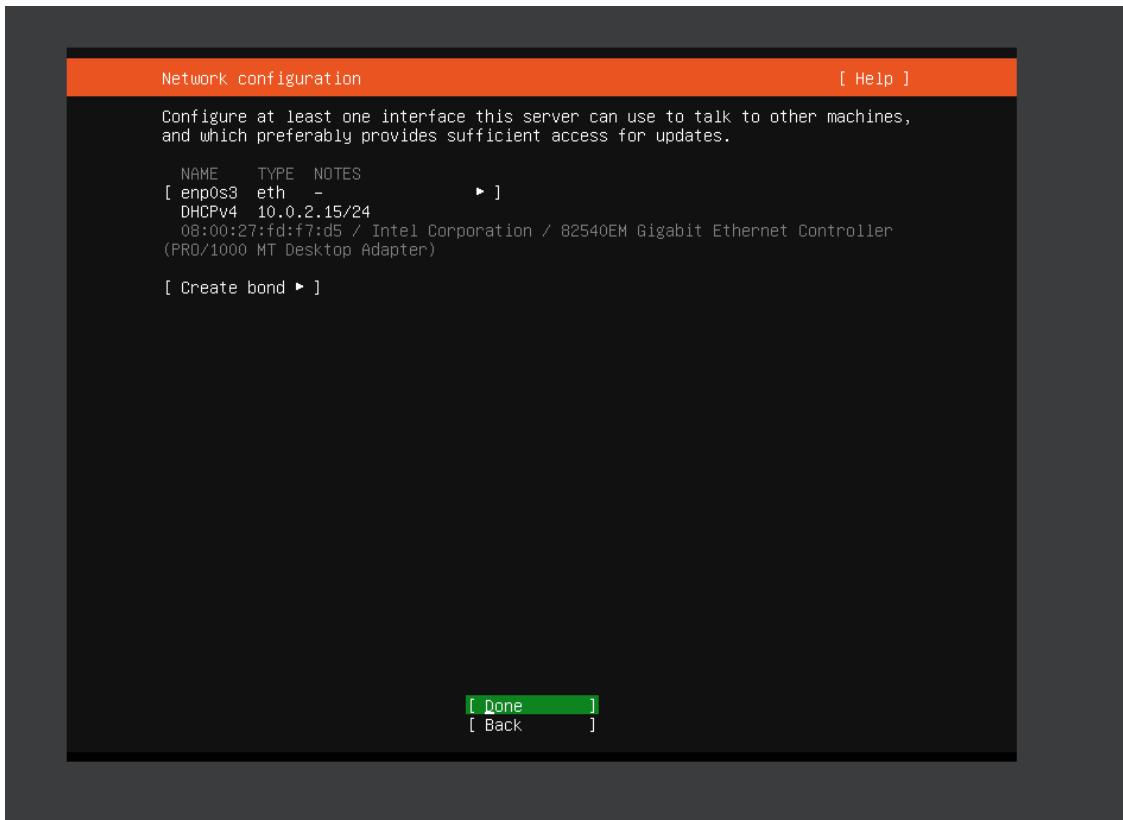


- Next will come Keyboard Configuration. Leave it as it is.

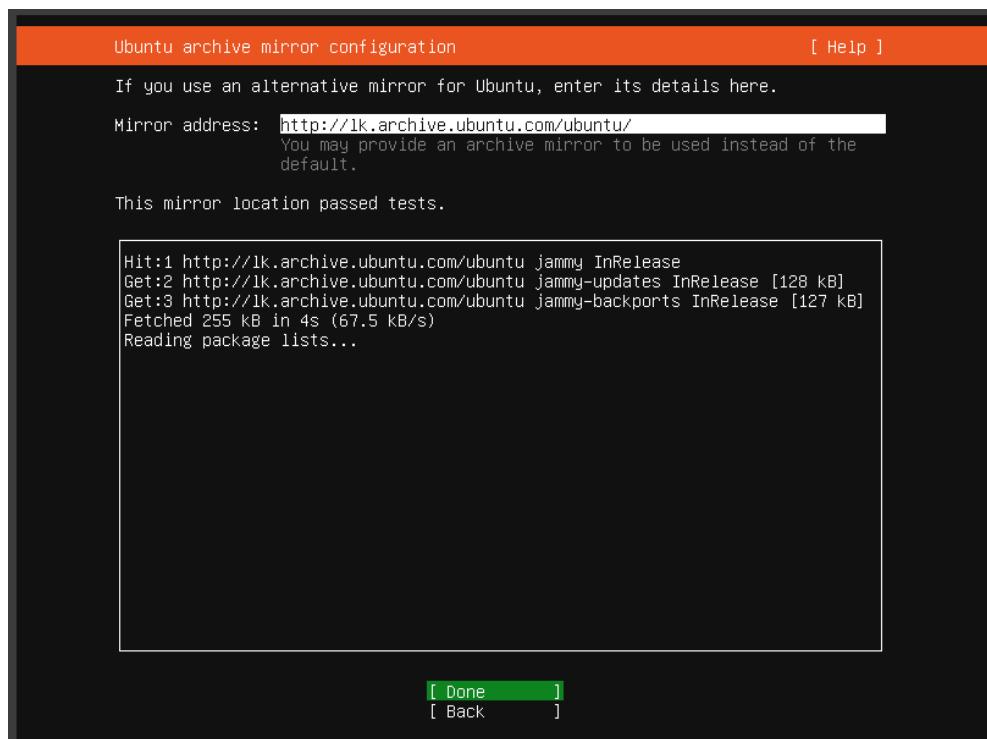
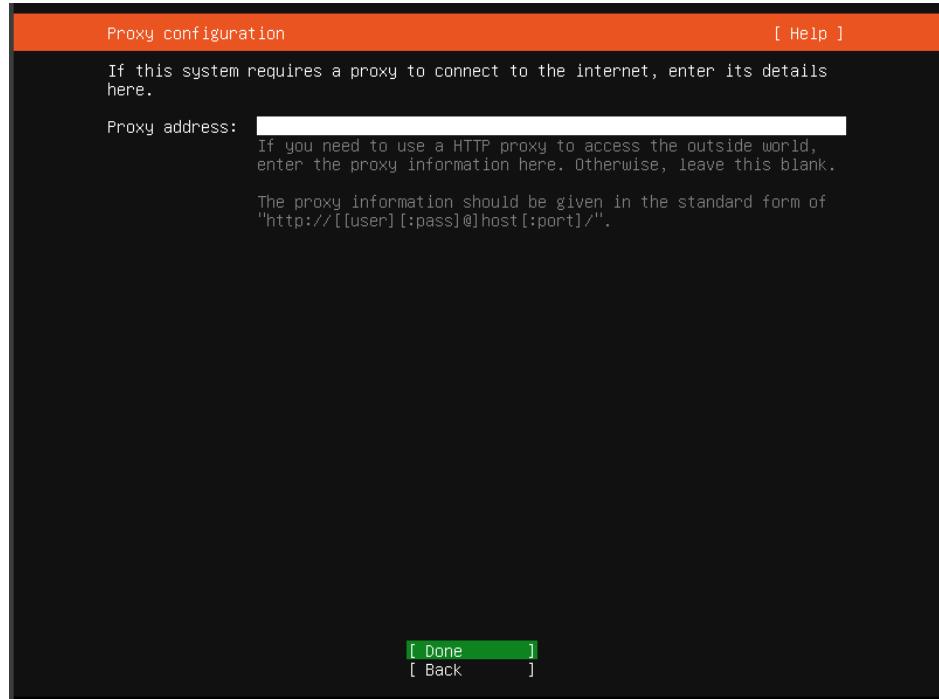




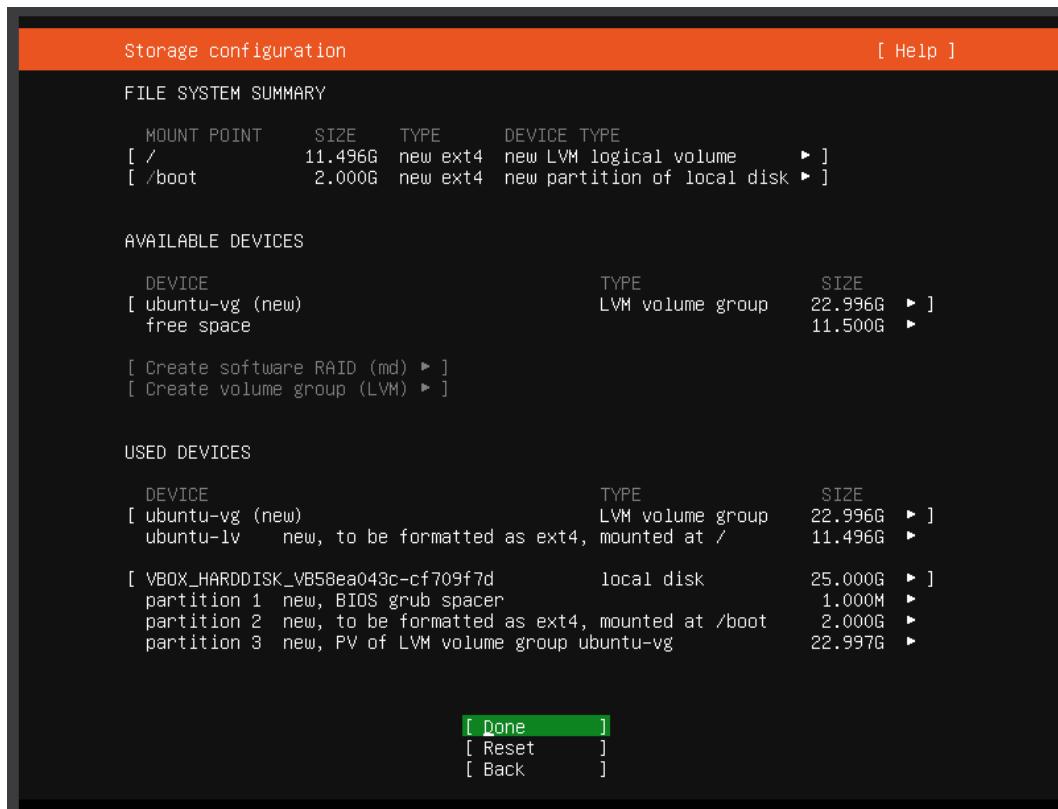
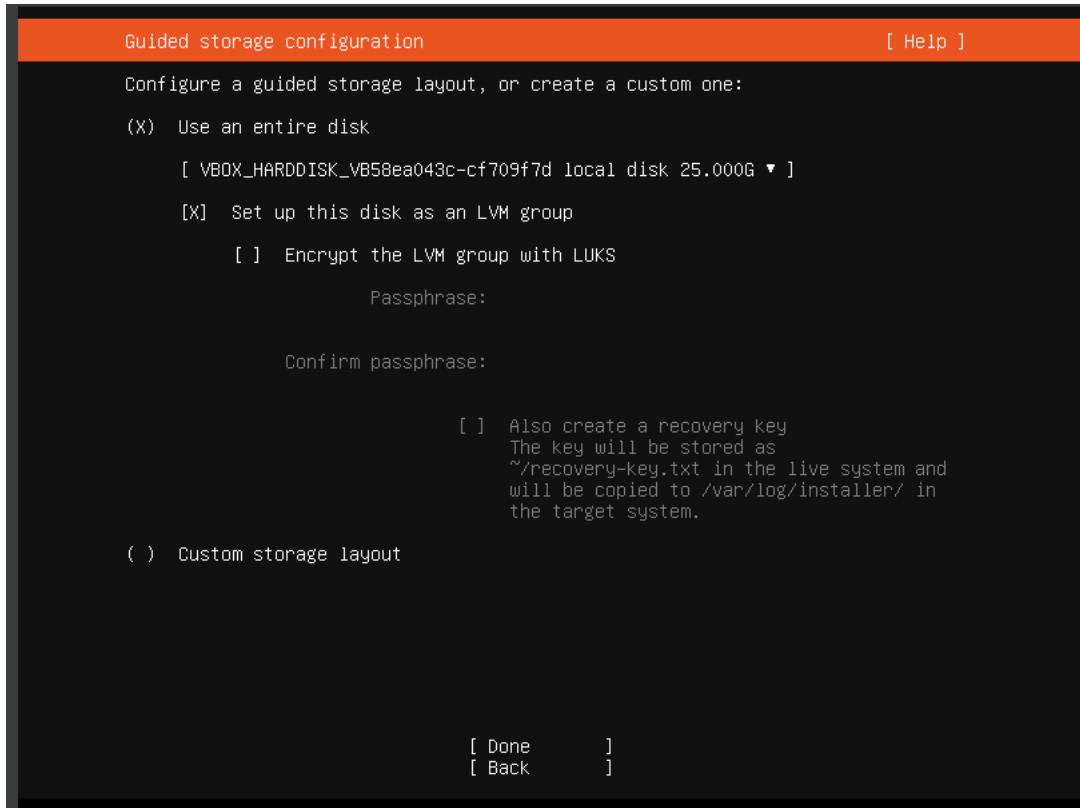
- Next, we will see the Network connections window. Let the server use DHCP to self-assign IP address to the machine.



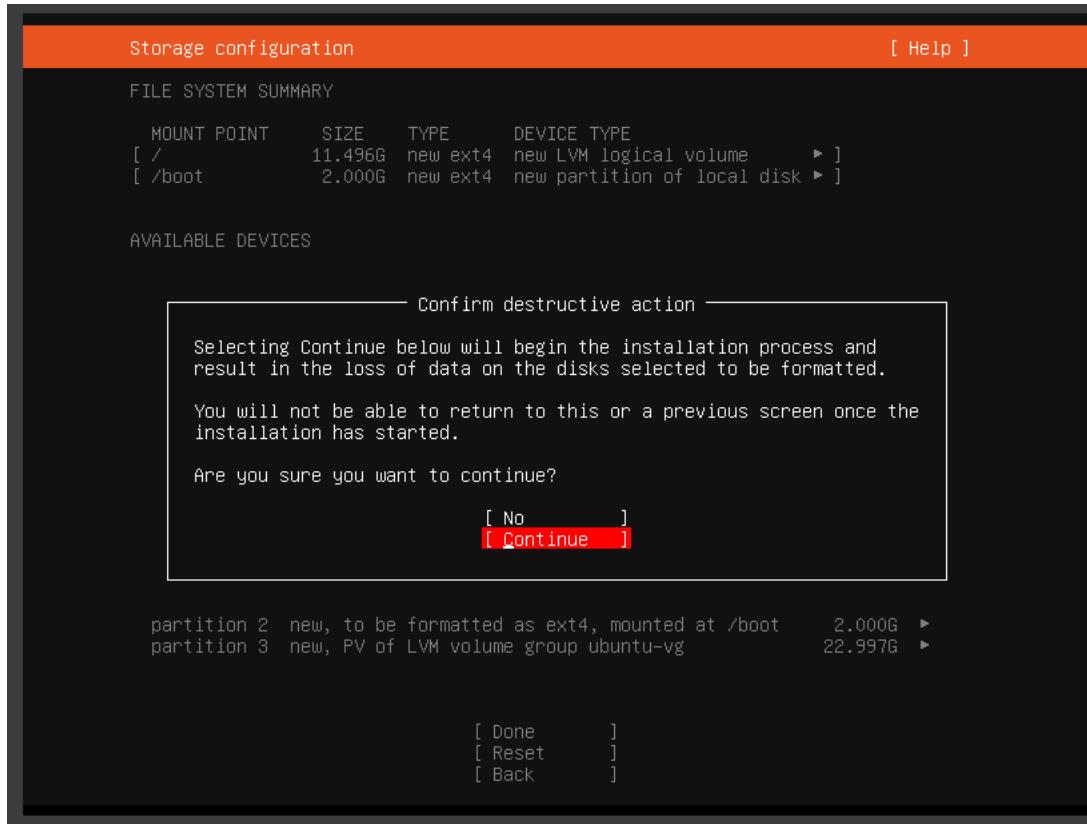
- Press Enter and now we will be presented to configure proxy but we'll skip it by pressing enter



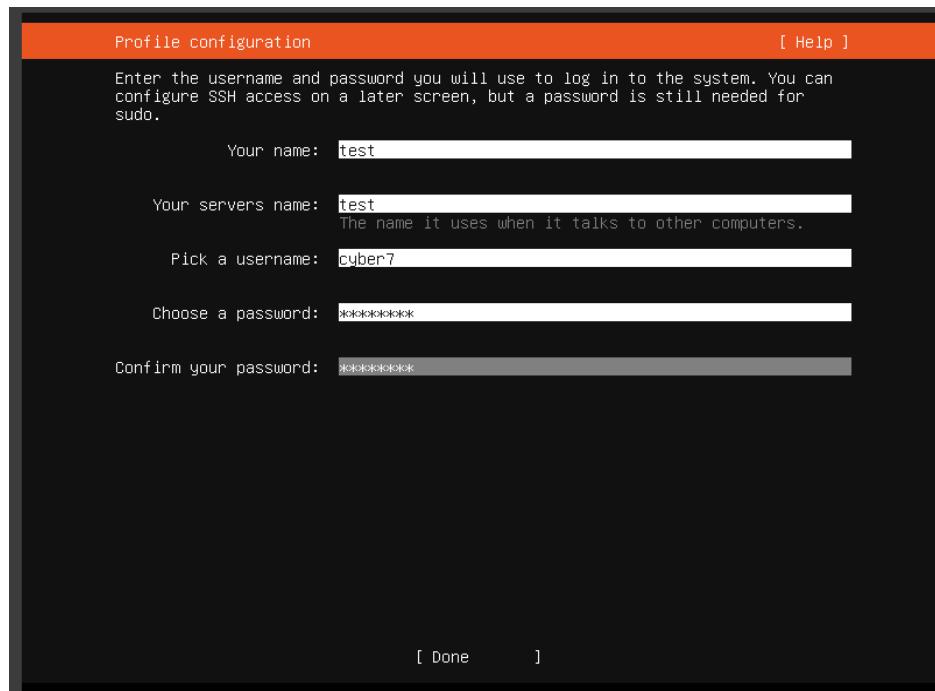
- Now, we will be presented with Storage configs. Use the entire disk for our box.



- Press Continue



- Next it will ask to enter the device name and a username and its corresponding password.



- Press Enter after providing name, username and password. Then it'll ask us if we want to install **OpenSSH Server**, and since we want to, so we'll select it by taking cursor to the area and pressing *spacebar*,

The screenshot shows a terminal window titled "Installing system". The log output is as follows:

```
subiquity/Zdev/apply_autoinstall_config:  
subiquity/Ad/apply_autoinstall_config:  
subiquity/Late/apply_autoinstall_config:  
configuring apt  
    curtin command in-target  
installing system  
executing curtin install initial step  
executing curtin install partitioning step  
    curtin command install  
    configuring storage  
        running 'curtin block-meta simple'  
    curtin command block-meta  
        removing previous storage devices  
        configuring disk: disk-sda  
        configuring partition: partition-0  
        configuring partition: partition-1  
        configuring format: format-0  
        configuring partition: partition-2  
        configuring lvm_volvgroup: lvm_volvgroup-0  
        configuring lvm_partition: lvm_partition-0  
        configuring format: format-1  
        configuring mount: mount-1  
        configuring mount: mount-0  
executing curtin install extract step  
    curtin command install  
        writing install sources to disk  
        running 'curtin extract'  
    curtin command extract  
        acquiring and extracting image from cp:///tmp/tmprh69oiuo/mount /
```

[ View full log ]

The screenshot shows a terminal window titled "Installation complete!". The log output is as follows:

```
curtin command in-target  
executing curtin install curthooks step  
    curtin command install  
        configuring installed system  
        running 'curtin curthooks'  
    curtin command curthooks  
        configuring apt  
        configuring apt  
        installing missing packages  
        Installing packages on target system: ['grub-pc']  
        configuring iscsi service  
        configuring raid (mdadm) service  
        configuring NVMe over TCP  
        installing kernel  
        setting up swap  
        apply networking config  
        writing etc/fstab  
        configuring multipath  
        updating packages on target system  
        configuring pollinate user-agent on target  
        updating initramfs configuration  
        configuring target system bootloader  
        installing grub to target devices  
        copying metadata from /cdrom  
final system configuration  
calculating extra packages to install  
configuring cloud-init  
restoring apt configuration  
subiquity/Late/run:
```

[ View full log ]  
[ Reboot Now ]

- Pressing Enter for the last time to finish with this configuration and ubuntu will start the installation process.
- After getting to this point, you will get an option to **Reboot Now**. Go ahead and reboot this machine and now our machine is ready.

## Implementation of Challenges

- Open the Ubuntu virtual machine
- Create the directories

```
cyber7@test:~$ sudo mkdir -p /var/www/csp-lab/templates_
cyber7@test:~$ tree /var/www/csp-lab/
/var/www/csp-lab/
├── app.py
└── templates
    └── index.html
        └── static

2 directories, 2 files
cyber7@test:~$ _
```

- Install Python3, Flask

```
cyber7@test:~$ sudo apt update
[sudo] password for cyber7:
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 c-n-f Metadata [30.3 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 c-n-f Metadata [488 B]
Get:7 http://archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2,266 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8,372 B]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,511 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [410 kB]
35% [10 Packages store 0 B] [11 Translation-en 95.6 kB/410 kB 23%] [8 Packages 1,300 kB/2,266 kB 57_
```

```
cyber7@test:~$ python3 --version
Python 3.10.12
cyber7@test:~$
```

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
^[[Bcyber7@test:~$ pip3 install flask_
```

```
systemctl restart unattended-upgrades.service  
systemctl restart user@1000.service  
  
No containers need to be restarted.  
  
No user sessions are running outdated binaries.  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
^[[Bcyber7@test:~$ pip3 install flask  
Defaulting to user installation because normal site-packages is not writeable  
Collecting flask  
  Downloading flask-3.1.0-py3-none-any.whl (102 kB)  
          103.0/103.0 KB 40.7 KB/s eta 0:00:00  
Collecting Werkzeug>=3.1  
  Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)  
          224.5/224.5 KB 380.0 KB/s eta 0:00:00  
Collecting blinker>=1.9  
  Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)  
Collecting Jinja2>=3.1.2  
  Downloading jinja2-3.1.6-py3-none-any.whl (134 kB)  
          134.9/134.9 KB 517.5 KB/s eta 0:00:00  
Collecting itsdangerous>=2.2  
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)  
Collecting click>=8.1.3  
  Downloading click-8.1.8-py3-none-any.whl (98 kB)  
          98.2/98.2 KB 574.2 KB/s eta 0:00:00  
Requirement already satisfied: MarkupSafe>=2.0 in /usr/lib/python3/dist-packages (from Jinja2>=3.1.2  
->flask) (2.0.1)  
Collecting MarkupSafe>=2.0  
  Downloading MarkupSafe-3.0.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (20 kB)  
Installing collected packages: MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, flask  
  WARNING: The script flask is installed in '/home/cyber7/.local/bin' which is not on PATH.  
    Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.  
Successfully installed Jinja2-3.1.6 MarkupSafe-3.0.2 Werkzeug-3.1.3 blinker-1.9.0 click-8.1.8 flask-  
3.1.0 itsdangerous-2.2.0  
cyber7@test:~$ _
```

```
inotify stopped and disabled on System startup  
cyber7@test:~$ cd /var/www/csp-lab/  
cyber7@test:/var/www/csp-lab$ sudo python3 app.py  
 * Serving Flask app 'app'  
 * Debug mode: on  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
 * Running on all addresses (0.0.0.0)  
 * Running on http://127.0.0.1:80  
 * Running on http://192.168.8.135:80  
Press CTRL+C to quit  
 * Restarting with stat  
 * Debugger is active!  
 * Debugger PIN: 380-534-068  
192.168.8.181 - - [27/Apr/2025 23:54:25] "GET / HTTP/1.1" 200 -  
192.168.8.181 - - [27/Apr/2025 23:54:25] "GET /favicon.ico HTTP/1.1" 404 -
```

```
cyber7@test:~$ cd /var/www/csp-lab/
cyber7@test:/var/www/csp-lab$ sudo nano /var/www/csp-lab/app.py
```

```
GNU nano 6.2                               /var/www/csp-lab/app.py
from flask import Flask, request, render_template, Response

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def home():
    return Response(render_template("index.html"), headers={
        "Content-Security-Policy": "default-src 'self'; script-src 'self' 'unsafe-inline'; >
    })

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80)
```

- Developing the vulnerable site

```
cyber7@test:~$ cd /var/www/csp-lab/templates/
cyber7@test:/var/www/csp-lab/templates$
```

```
cyber7@test:/var/www/csp-lab/templates$ sudo nano index.html
```

```
GNU nano 6.2                               index.html *
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSP Misconfigurations Demo</title>
    <style>
        body{
            font-family: Arial, sans-serif;
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
            background-color: #f0f0f0;
        }
        h1{
            color: #333;
            text-align: center;
        }
        .challenge{
            background-color: white;
            padding: 20px;
            margin-bottom: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 4px rgba(0,0,0,0.1);
        }
        .challenge h2{
            color: #2c3e50;
        }
        textarea{
            width: 100%;
            height: 100px;
            margin-bottom: 10px;
            padding: 10px;
        }
    </style>
```

```
GNU nano 6.2                               index.html *
textarea{
    width: 100%;
    height: 100px;
    margin-bottom: 10px;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
}
button{
    padding: 10px 20px;
    background-color: #3498db;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
#comments{
    margin-top: 20px;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
    min-height: 100px;
}
.config-links{
    margin-bottom: 20px;
}
.config-links a{
    margin-right: 10px;
    text-decoration: none;
    color: #3498db;
}
.config-links a:hover{
    color: #2980b9;
}

^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location   M-U Undo
^X Exit      ^R Read File   ^Y Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo
```

```
GNU nano 6.2                               index.html *
.config-links a:hover{
    color: #2980b9;
}

```

```
</style>
</head>
<body>
    <h1>CSP Misconfigurations Demo</h1>
    <div class="config-links">
        <a href="/unsafe-inline">Challenge 1: Unsafe-inline</a>
        <a href="/permissive">Challenge 2: Permissive script-src</a>
        <a href="/nonce">Challenge 3: Missing Nonce Validation</a>
    </div>
    <div class="challenge">
        <h2>Comment Section</h2>
        <p>Enter a comment below. Try to inject XSS Payloads!</p>
        <textarea id="commentInput" placeholder="Enter your comment..."></textarea>
        <button onclick="submitComment()">Submit Comment</button>
        <div id="comments"></div>
    </div>
    <script>
        function submitComment(){
            const input = document.getElementById('commentInput').value;
            const commentsDiv = document.getElementById('comments');
            const commentElement = document.createElement('div');
            commentElement.innerHTML = input;
            commentsDiv.appendChild(commentElement);
            document.getElementById('commentInput').value = '';
        }
    </script>
</body>
</html>
```

```
^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location   M-U Undo
^X Exit      ^R Read File   ^Y Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo
```

## Downloaded File Challenge 1

- Create a project file

```
File Actions Edit View Help
└─(kali㉿kali)-[~]
  $ mkdir csp-misconfig-challenge
  cd csp-misconfig-challenge
```

- Create a nano file to create a web page

```
└─(kali㉿kali)-[~/csp-misconfig-challenge]
  $ nano index.html
```

```
GNU nano 8.2
index.html *

</style>
</head>
<body>
  <h1>Can You Bypass the CSP?</h1>
  <p>Inject your payload and try to make the flag appear.</p>
  <input type="text" id="payload" placeholder="Enter your CSP bypass payload">
  <br>
  <button onclick="runPayload()">Run Payload</button>
  <div id="result"></div>
  <script>
    function runPayload() {
      const input = document.getElementById("payload").value;
      document.getElementById("result").innerHTML = input;
    }
  </script>
  <div class="hint">
    Hint: The CSP allows scripts from <code>https://evil.tryhackme.site</code>
  </div>
</body>
</html>
```

- Create a JavaScript file that would simulate an attack-controlled domain

```
File System
└─(kali㉿kali)-[~/csp-misconfig-challenge]
  $ nano payload.js
```

```
File Actions Edit View Help
payload.js *

GNU nano 8.2
alert("THM{csp_bypassed_flag}");
```

## Downloaded File Challenge 2

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Vulnerable Web App - TryHackMe Task</title>
7      <style>
8          body {
9              font-family: Arial, sans-serif;
10             margin: 40px;
11             background-color: #f0f0f0;
12         }
13         .container {
14             max-width: 800px;
15             margin: 0 auto;
16             background-color: white;
17             padding: 20px;
18             border-radius: 8px;
19             box-shadow: 0 0 10px rgba(0,0,0,0.1);
20         }
21         h1 {
22             color: #333;
23         }
24         .section {
25             margin-bottom: 20px;
26         }

```

▲ htdocs/index.html

```

25     margin-bottom: 20px;
26 }
27 input, button {
28     padding: 8px;
29     margin: 5px 0;
30 }
31 #output {
32     margin-top: 10px;
33     padding: 10px;
34     border: 1px solid #ddd;
35 }
36 .hint {
37     color: #555;
38     font-style: italic;
39 }
40 <!-- Hidden comment with a clue for users -->
41 <!-- Hint: Check the page source for sensitive information. Password: tryhackme123 -->
42 </style>
43 </head>
44 <body>
45     <div class="container">
46         <h1>Vulnerable Web App - TryHackMe Exploitation Task</h1>
47
48         <div class="section">
49             <h2>Task 1: Sensitive Data Exposure</h2>
50             <p>Inspect the page source code to find hidden information, such as a password. Right-click and select "View Page Source" to start.</p>

```

▲ htdocs/index.html

```

51         <p class="hint">Hint: Developers sometimes leave sensitive data in comments.</p>
52     </div>
53
54     <div class="section">
55         <h2>Task 2: HTML Injection</h2>
56         <p>Enter your name below. The input is displayed on the page, but it's not sanitized. Can you inject HTML to create a malicious link?</p>
57         <form id="nameForm">
58             <label for="nameInput">Your Name:</label><br>
59             <input type="text" id="nameInput" placeholder="Enter your name">
60             <button type="submit">Submit</button>
61         </form>
62         <div id="output"></div>
63         <p class="hint">Hint: Try injecting HTML like &lt;a href="http://hacker.com"&gt;Click Me&lt;/a&gt;.</p>
64     </div>
65
66     <div class="section">
67         <h2>Task 3: File Upload Vulnerability</h2>
68         <p>Upload a file to the server. The server doesn't validate file types properly. Can you upload a malicious file?</p>
69         <form id="uploadForm" enctype="multipart/form-data">
70             <label for="fileInput">choose a file:</label><br>
71             <input type="file" id="fileInput">
72             <button type="submit">Upload</button>
73         </form>
74         <div id="uploadOutput"></div>
75         <p class="hint">Hint: Try uploading a file with a .php extension disguised as an image (e.g., shell.php.jpg).</p>
76     </div>

```

▲ htdocs/index.html

```
72 |     <button type="submit">Upload</button>
73 |   </form>
74 |   <div id="uploadOutput"></div>
75 |   <p class="hint">Hint: Try uploading a file with a .php extension disguised as an image (e.g., shell.php.jpg).</p>
76 | </div>
77 |</div>
78 |
79 <script>
80 // Task 2: HTML Injection
81 document.getElementById('nameForm').addEventListener('submit', function(e) {
82     e.preventDefault();
83     const name = document.getElementById('nameInput').value;
84     // Vulnerable: Directly injecting user input without sanitization
85     document.getElementById('output').innerHTML = `Hello, ${name}!`;
86     // Check for successful HTML injection
87     if (name.includes('hacker.com')) {
88         alert('Flag: HTML_INJECTION - You successfully injected a malicious link!');
89     }
90 });
91
92 // Task 3: File Upload Simulation
93 document.getElementById('uploadForm').addEventListener('submit', function(e) {
94     e.preventDefault();
95     const inputFile = document.getElementById('fileInput');
96     if (inputFile.files.length > 0) {
97         const fileName = inputFile.files[0].name.toLowerCase();
```

▲ htdocs/index.html

## References

1. "How to make our own CTF Challenge with ease.", Medium, 2022. [Online]. Available: <https://infosecwriteups.com/how-to-make-our-own-ctf-challenge-with-ease-6b15f76865b5>. [Accessed: 5-Dec- 2022].
2. "Create a CTF: Capture the Flag box!", Reddit.com, 2022. [Online]. Available: [https://www.reddit.com/r/cybersecurity/comments/fqjkg1/create\\_a\\_ctf\\_capture\\_the\\_flag\\_box/](https://www.reddit.com/r/cybersecurity/comments/fqjkg1/create_a_ctf_capture_the_flag_box/). [Accessed: 5-Dec- 2022].
3. "OWASP Top 10:2021", Owasp.org, 2022. [Online]. Available: <https://owasp.org/Top10/>. [Accessed: 5-Dec- 2022].
4. <https://www.imperva.com/learn/application-security/content-security-policy-csp-header/>