| | |
|---|---|
| **Vulnerability Name:** | Stored Cross Site Scripting (XSS) |
| **Affected Vendor:** | DVWA |
| **Affected Product Name:** | http://dvwa/vulnerabilities/xss_s/ |
| **Product Official Website URL:** | http://dvwa/login.php |
| **Affected Component:** | Comment box |

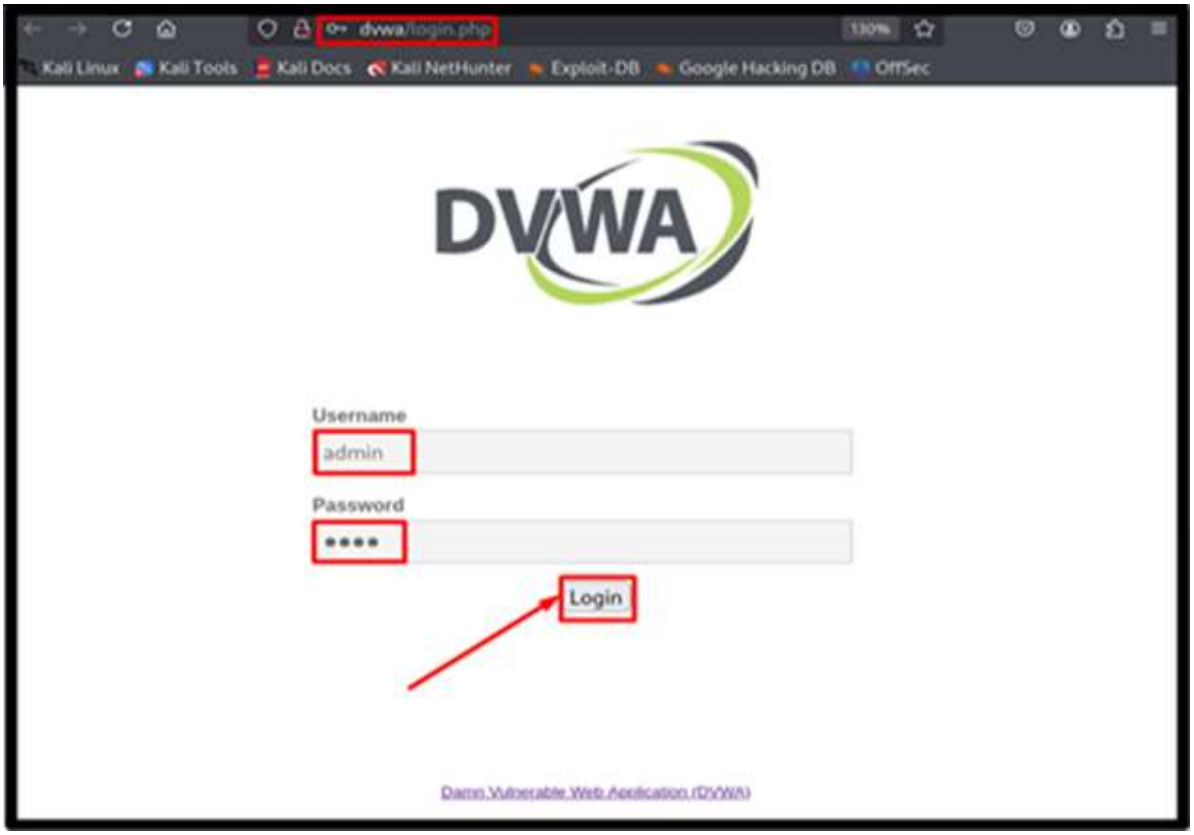
**Description:** Stored Cross-Site Scripting (XSS) is a cybersecurity vulnerability in which attackers inject harmful scripts into a web application. Unlike reflected XSS, the injected script is permanently stored on the server, often in a database, comment section, user profile, or other persistent storage mechanisms. When other users or administrators access the affected page, the malicious script is included in the server's response and executed within their browsers.

**Root Cause: -** Stored XSS occurs when user input is saved in a database and later displayed without proper sanitization, allowing malicious scripts to execute in users' browsers.

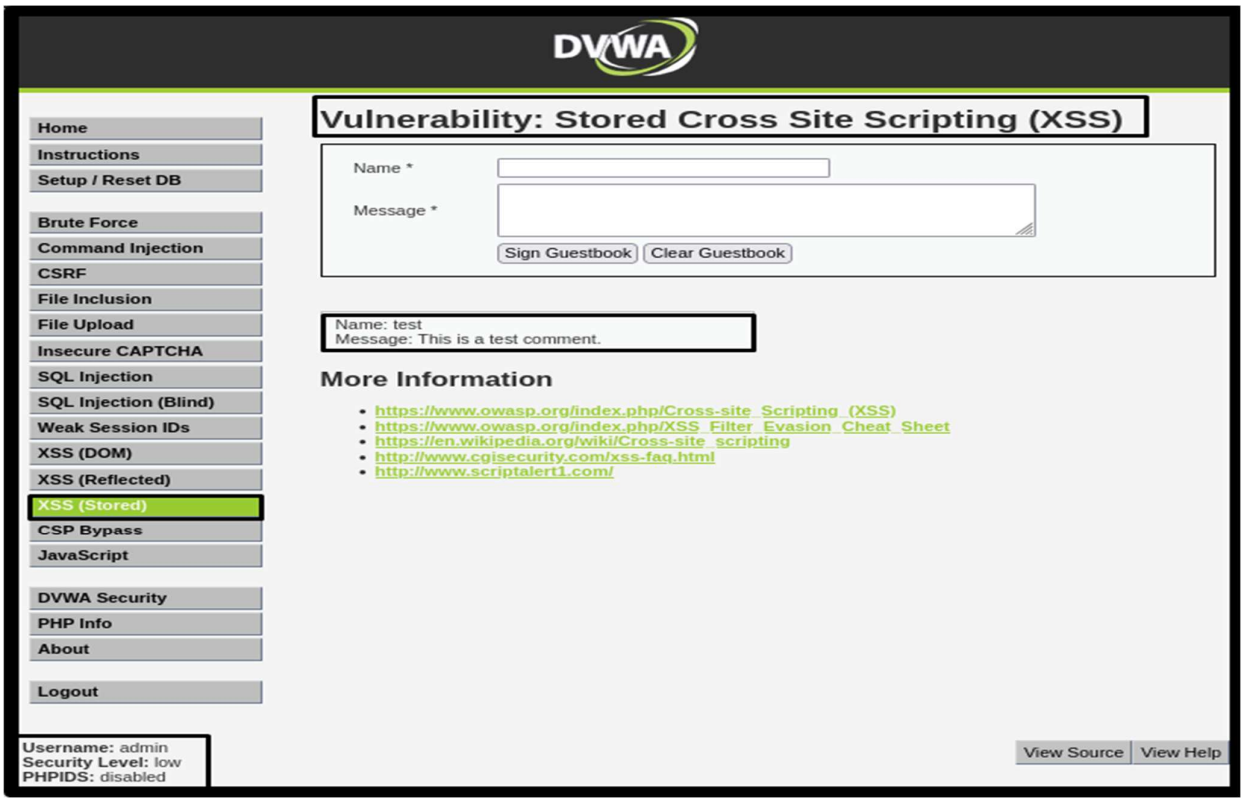
**Mitigation: -** Input Validation – Restrict allowed characters. Output Encoding – Encode data before rendering. CSP (Content Security Policy) – Block unauthorized scripts. Sanitize User Input – Remove harmful scripts before storing. use Prepared Statements – Prevent injection attacks.
**Impact:** The impact of Stored Cross-Site Scripting can range from stealing sensitive information, such as login credentials or personal data, to performing actions on behalf of multiple users without their consent. Additionally, attackers can use Stored Cross-Site Scripting to deliver malware, deface web pages, conduct phishing attacks, or compromise administrative accounts, potentially causing widespread reputational and operational damage to the affected organization.

**Remediation**: To remediate Stored Cross-Site Scripting (XSS), developers must adopt secure coding practices, including input validation, output encoding, and implementing a strict Content Security Policy (CSP). Additionally, regular security audits, penetration testing, and the utilization of web application firewalls (WAFs) are essential to detect and mitigate XSS vulnerabilities effectively. educating developers on secure development and continuously monitoring application behaviour also play a crucial role in preventing stored XSS attacks.

**Step: -1** First navigate to http://dvwa/login.php and login with username and Password.

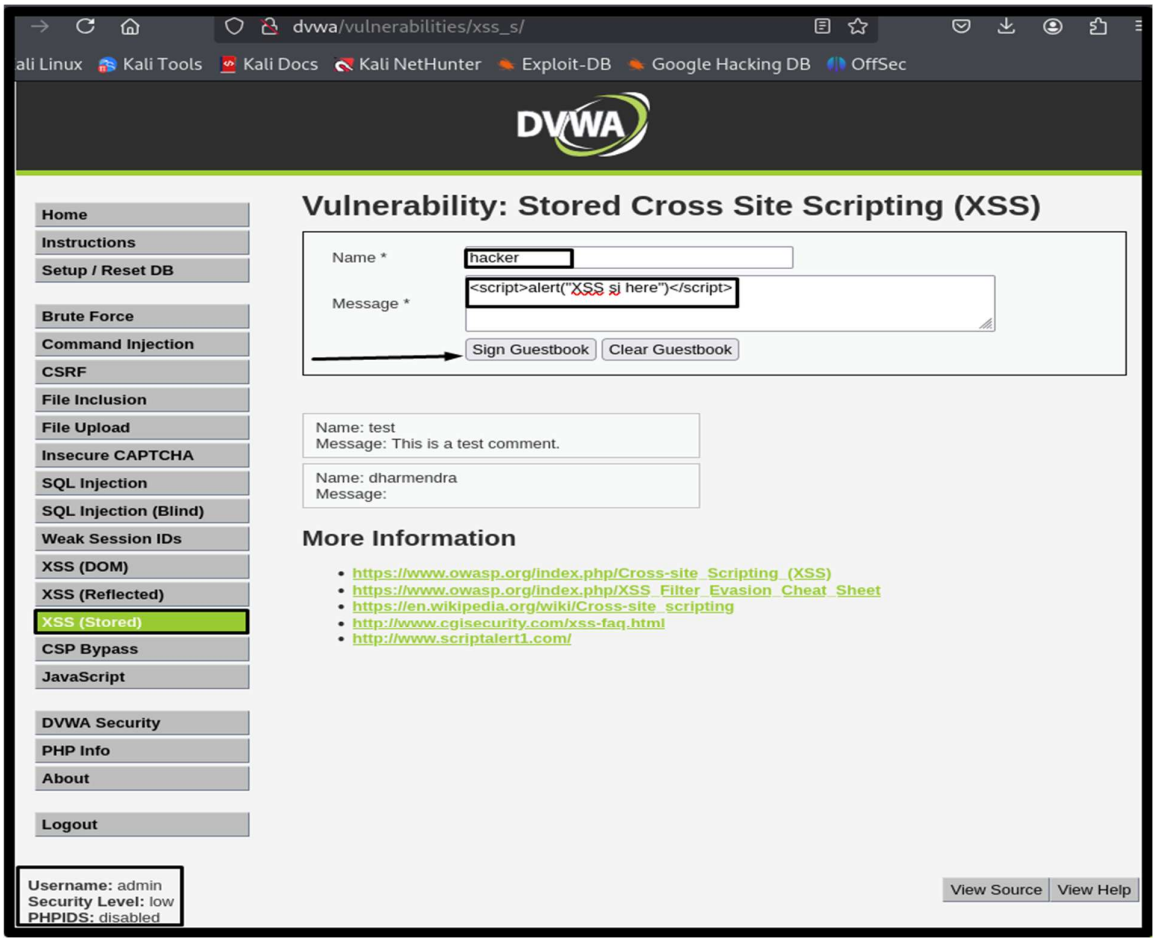



**Step: -2** log in the home page of DVWA then click to the Stored Cross Site Scrip ng (XSS) Section.

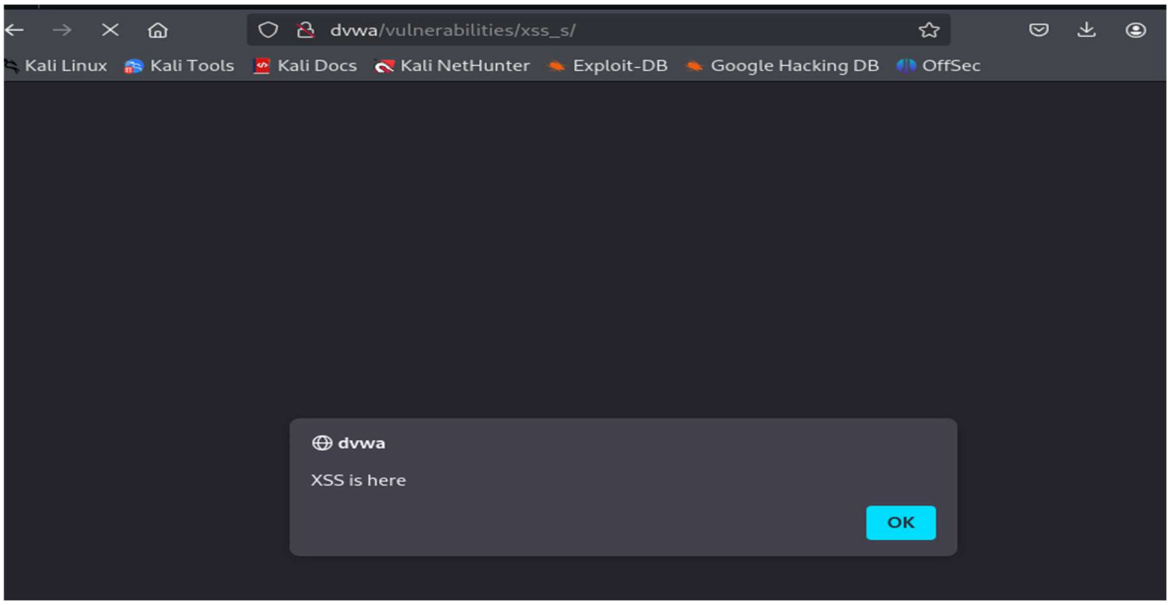


## SECURITY LEVEL (Low)

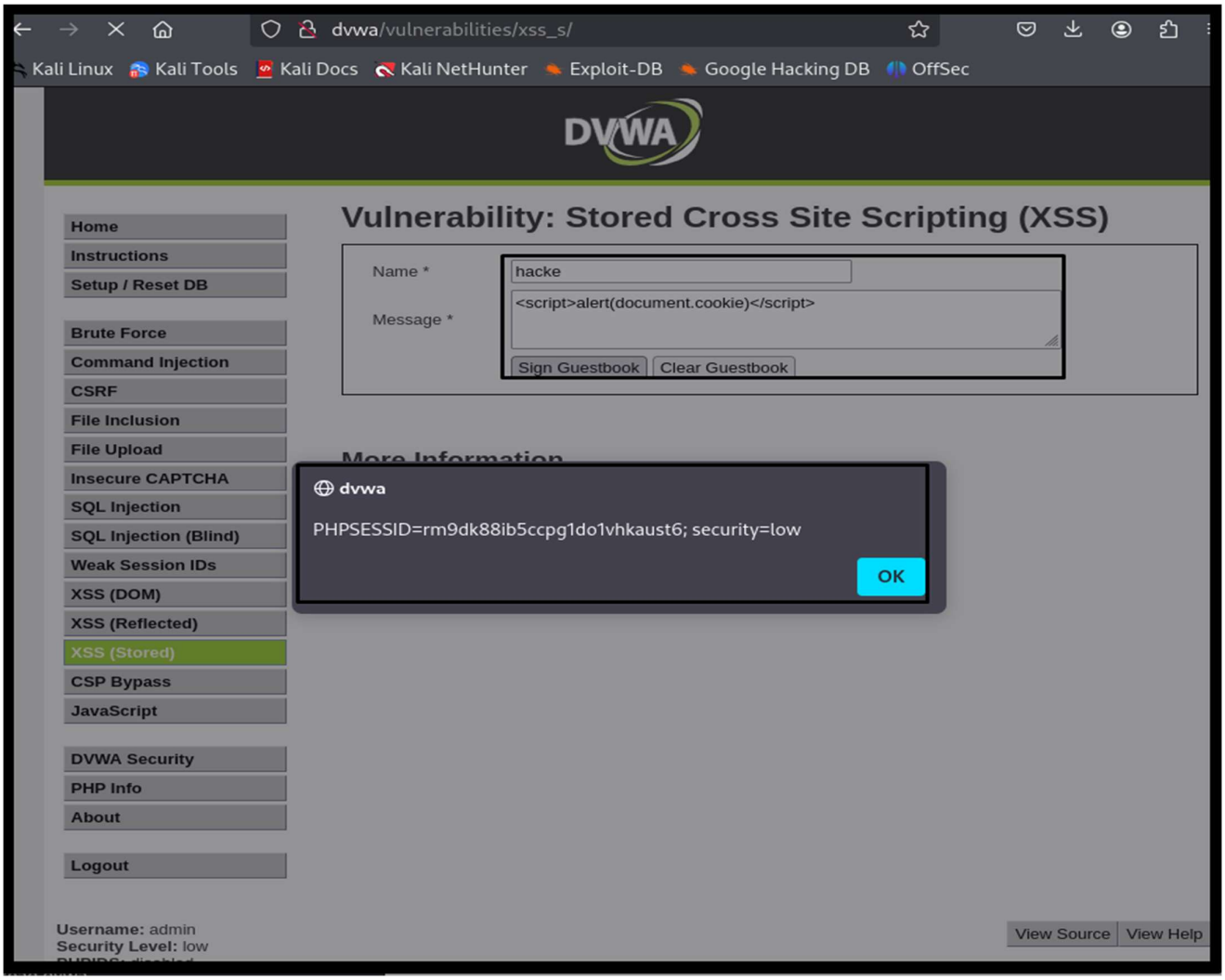
**Step: -3** In this Step I am fill the the all parameters and the apply a javascript payloads <script>alert(“XSS is here”)</script> and then click to Sign Guestbook and reload the page.




**Step: -** Show the pop

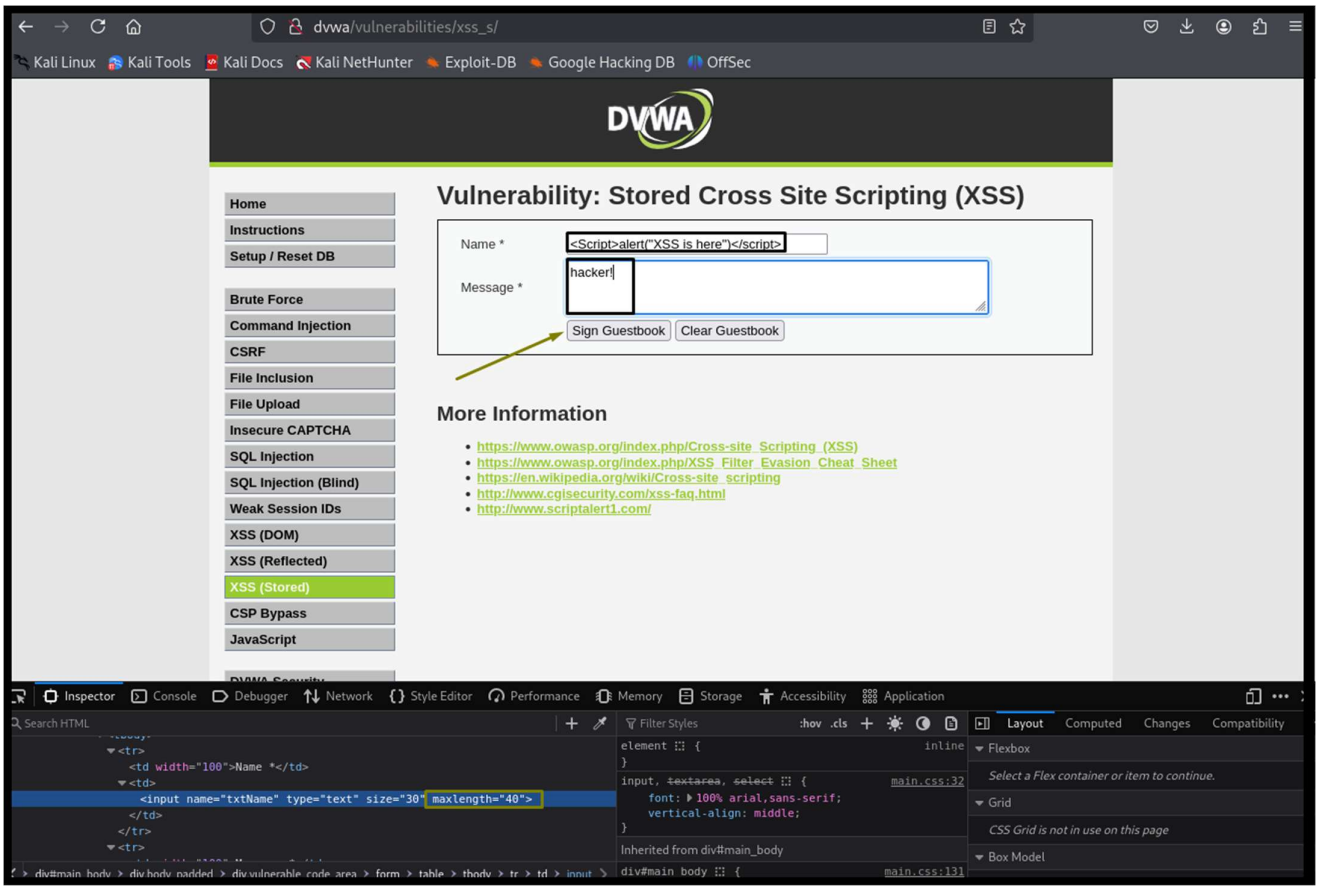


**Step: - 4** In this step, I am finding the cookie, and the payload is: <script>alert(document.cookie)</script>

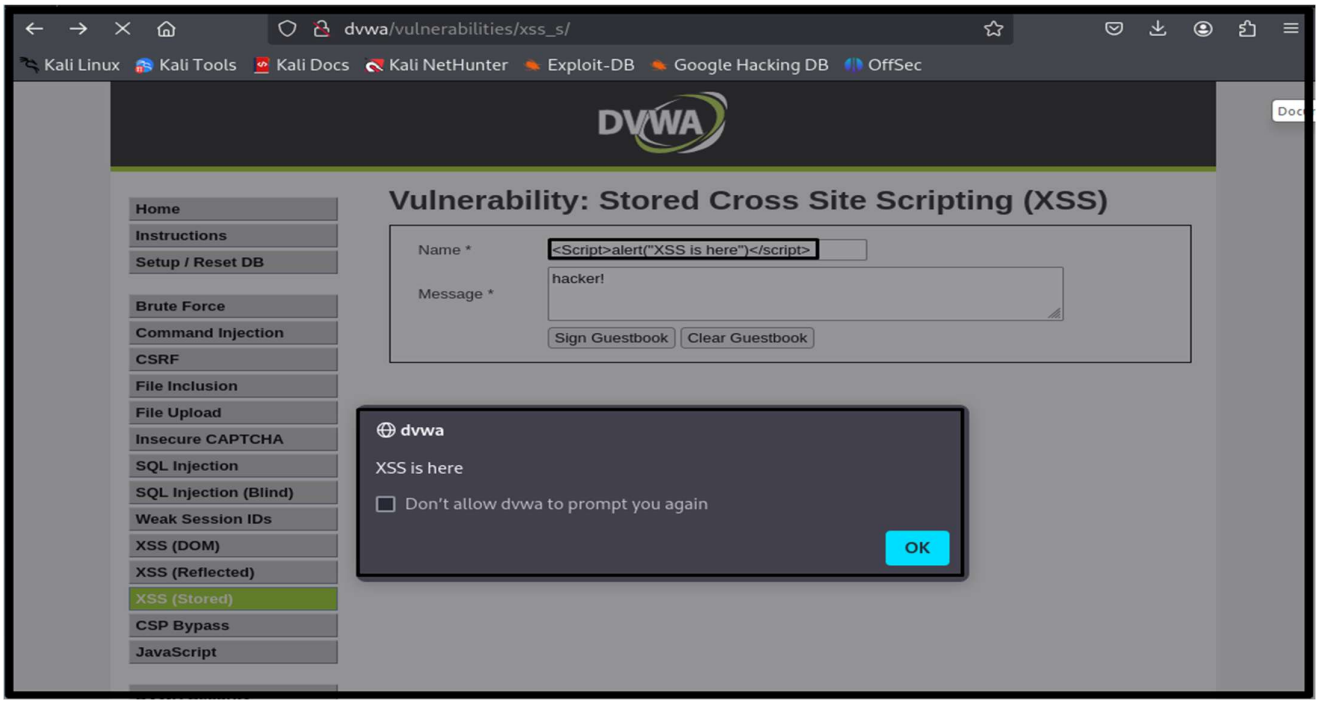


## SECURITY LEVEL (MEDIUM)

**Step: -1** In this Step firstly I changed max length10 to 40 then the name parameters apply in this payloads <sCript>alert("XSS is here")</script>

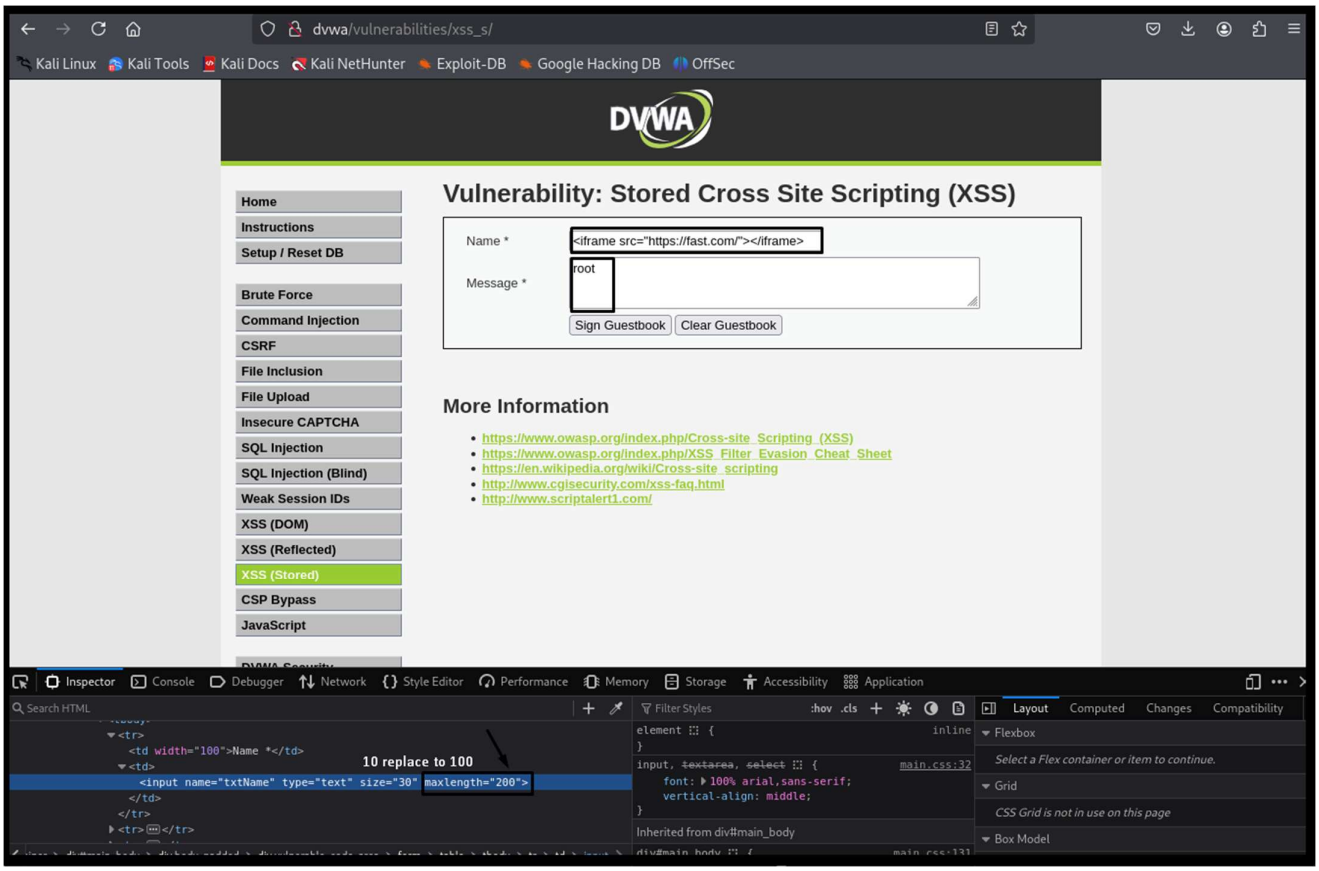



**Step: -2** Show the pop

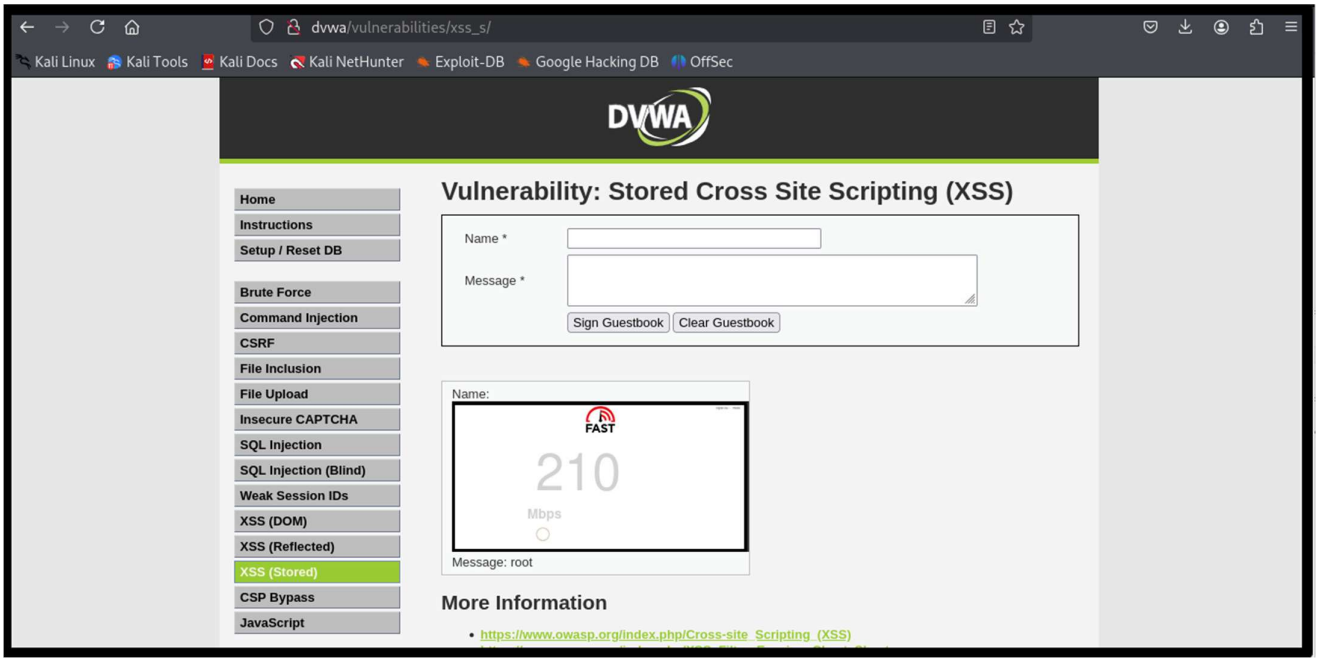


## SECURITY LEVEL (HIGH)

**Step: -1** In this Step firstly I changed max length10 to 40 then the name parameters apply in this payloads <iframe src=”https.com/”><iframe>

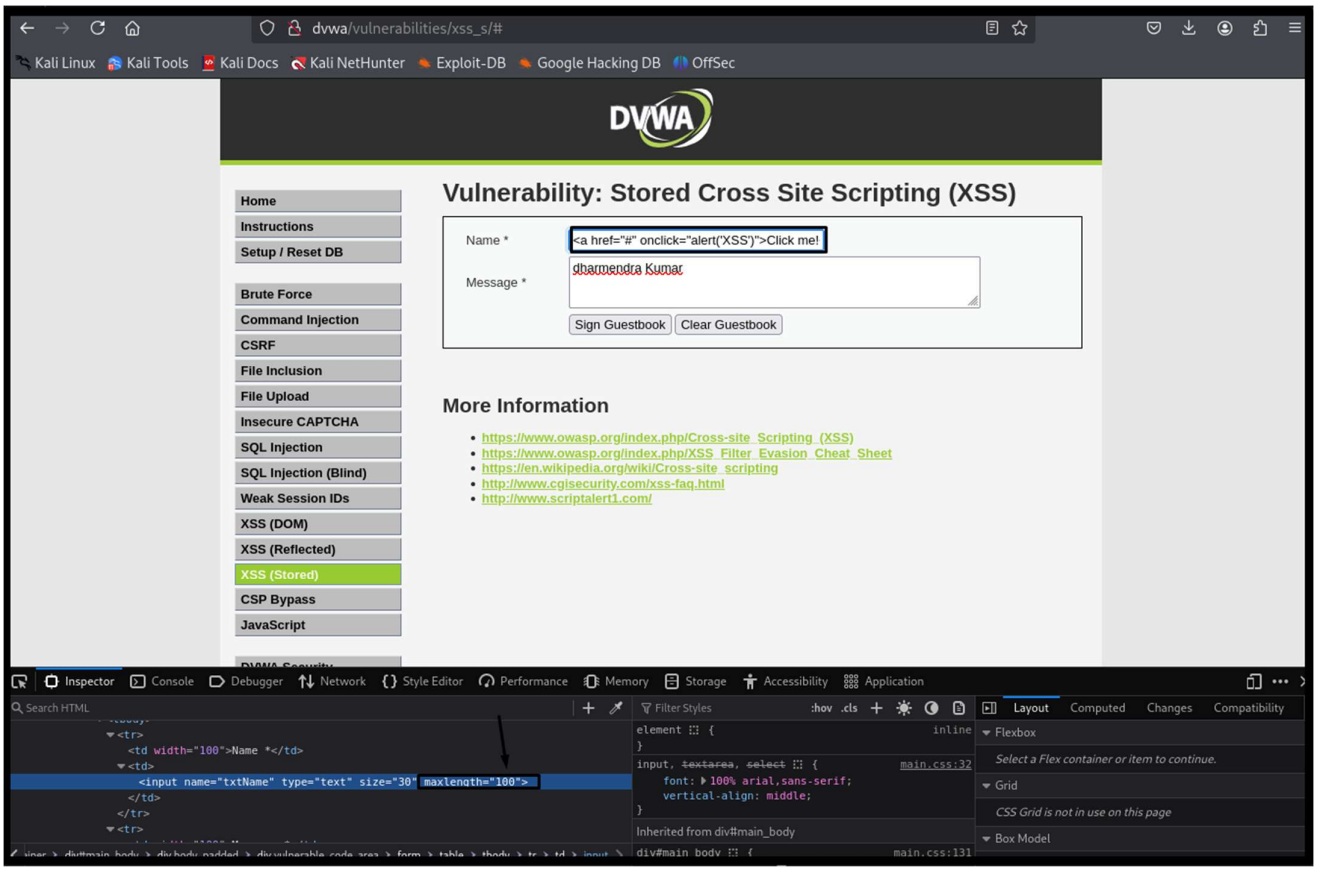



**Step: - 2** See the page

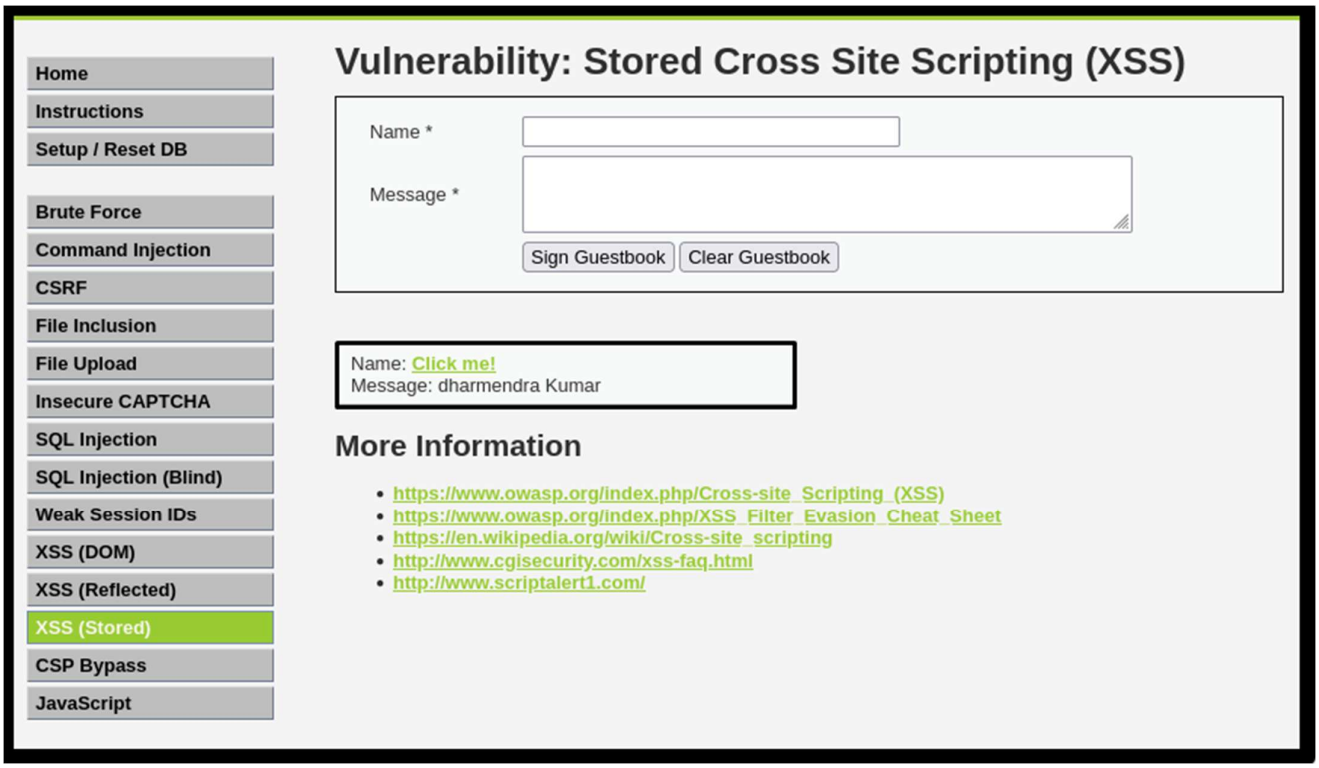


## 2nd method

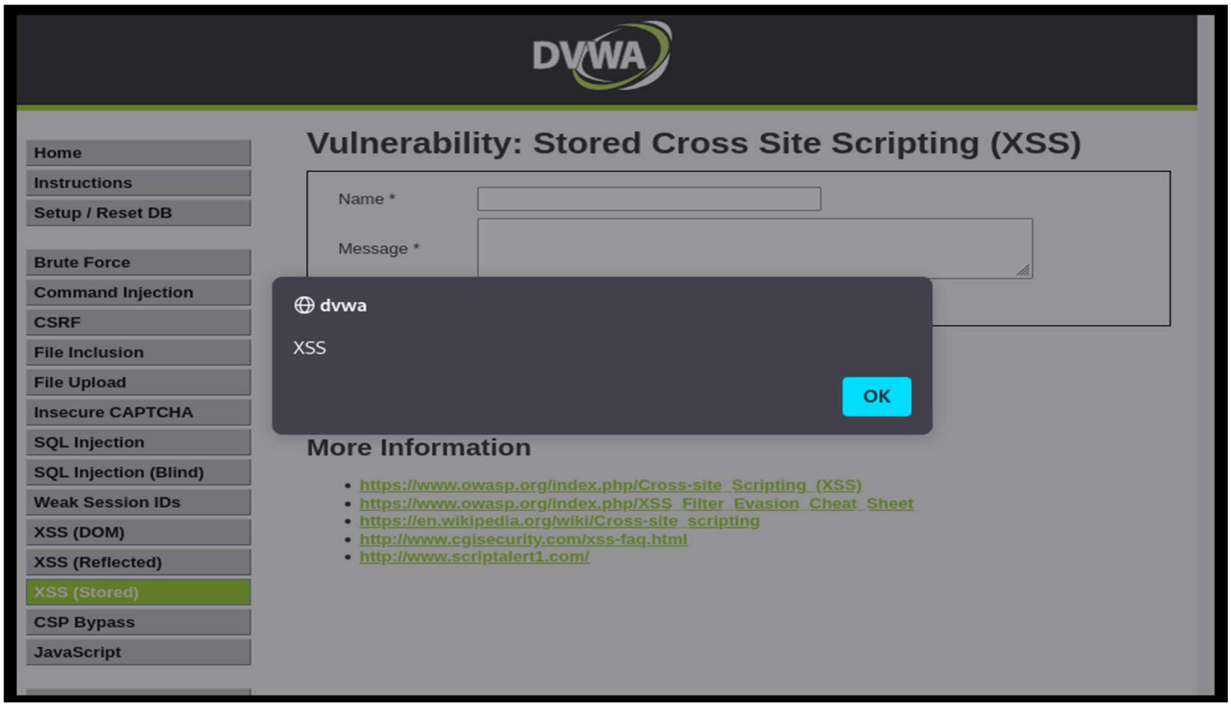
**Step: -1** In this Step firstly I changed max length10 to 100 then the name parameters apply in this payloads: <a href-“#” onclick=”alert(‘XSS’)”>Click me!




**Step: -2** In this Step Click to click me!



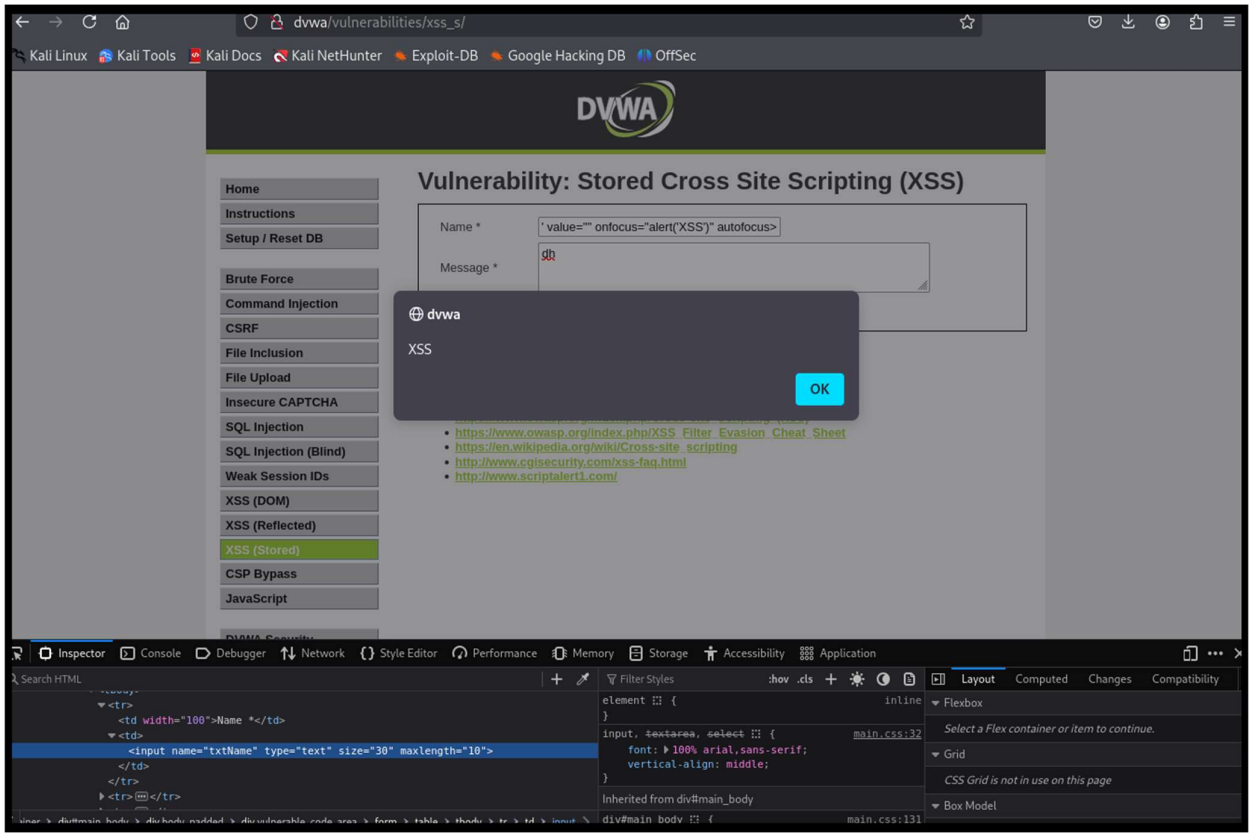

**Step: - 3** Show the pop




**Step: - 1** In this Step firstly I changed max length10 to 100 then the name parameters apply in this payloads:

<input type="text"value=""onfocus="alert('XSS')" autofocus>

Then show the pop.


dvwa/vulnerabilities/xss_s/
Kali Linux
Kali Tools
Kali Docs
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
' value="" onfocus="alert('XSS')" autofocus>
Message *
dh
dvwa
XSS
OK
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
http://www.scriptalert1.com/
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Search HTML
<td width="100">Name *</td>
<input name="txtName" type="text" size="30" maxlength="10">
Filter Styles
element { inline
input, textarea, select { main.css:32
font: 100% arial,sans-serif;
vertical-align: middle;
Inherited from div#main_body
div#main_body { main.css:131
Layout
Computed
Changes
Compatibility
Flexbox
Select a Flex container or item to continue.
Grid
CSS Grid is not in use on this page
Box Model