Assignment 4

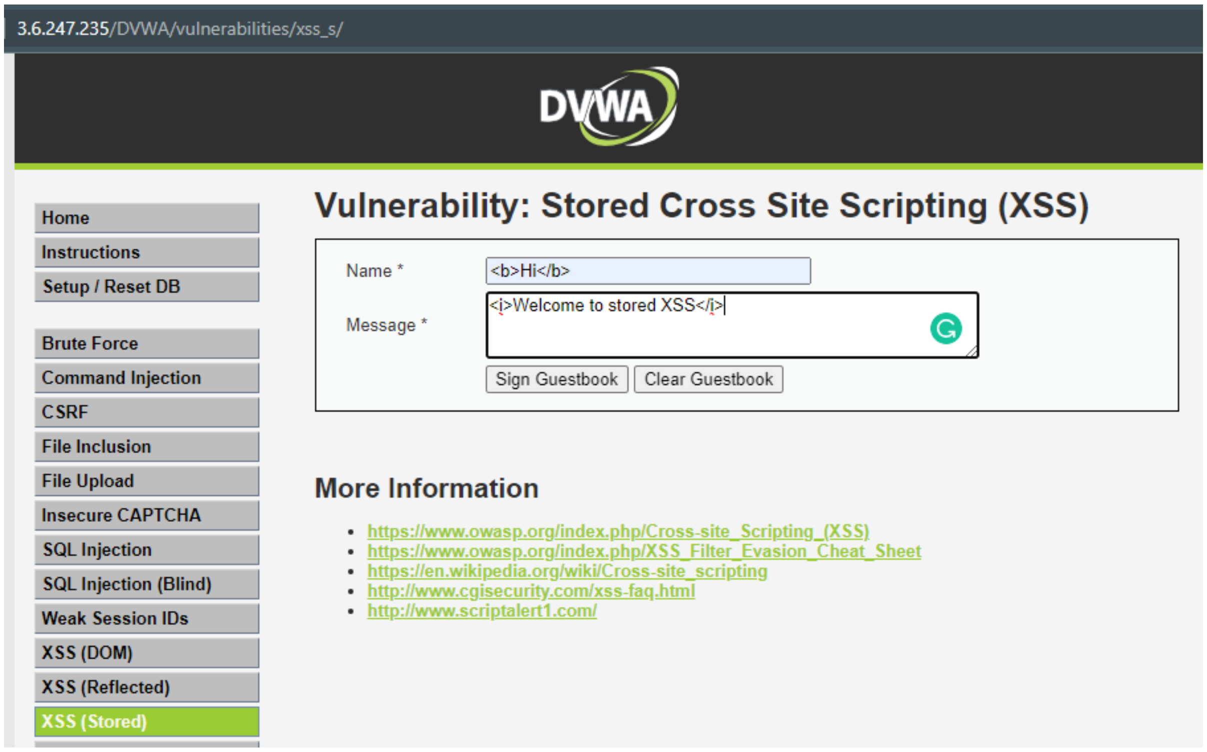
Ruchita Dinesh Entoliya (015331966)
Komal Joshi (014682551)

Stored XSS attack

1. **Describe the attack you used. How did it work?**

Stored XSS is the most dangerous cross site vulnerability. This type of vulnerability arises whenever a web application stores user supplied data for later use in backend without performing any filter or input sanitization. Since the web application does not apply any filter therefore an attacker can inject some malicious code into this input field.

After navigating to XSS stored tab, on entering html tags in name and message field we see html injection takes place where tags modify the responses.


3.6.247.235/DVWA/vulnerabilities/xss_s/

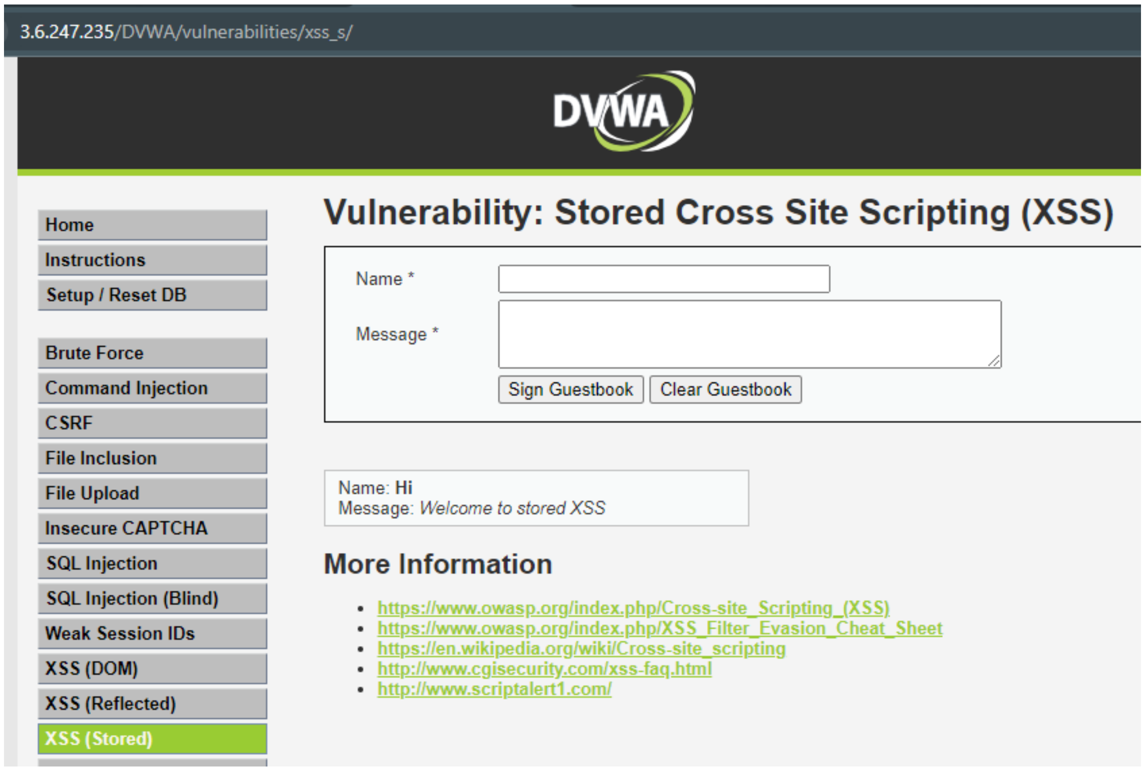
Vulnerability: Stored Cross Site Scripting (XSS)

Name * `<b>Hi</b>`

Message * `<i>Welcome to stored XSS</i>`

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- 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/


3.6.247.235/DVWA/vulnerabilities/xss_s/

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

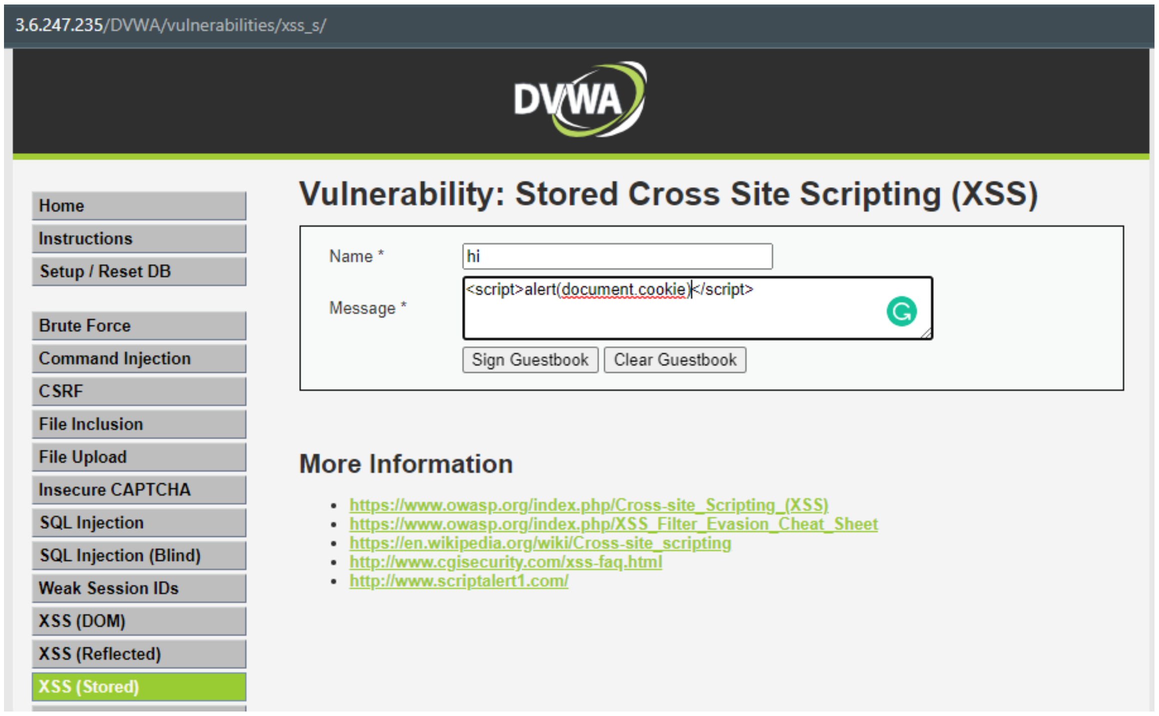
Name: Hi
Message: *Welcome to stored XSS*

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- 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/

To check JavaScript injection, entering <script>alert("JavaScript!")</script> in message field causes JavaScript injection. Alert message appears on the screen. On low security, page is vulnerable to XSS.

Next, we ty to get session ID and cookie to display using document.cookie as the alert parameter within the script tag in message field.


3.6.247.235/DVWA/vulnerabilities/xss_s/

Vulnerability: Stored Cross Site Scripting (XSS)

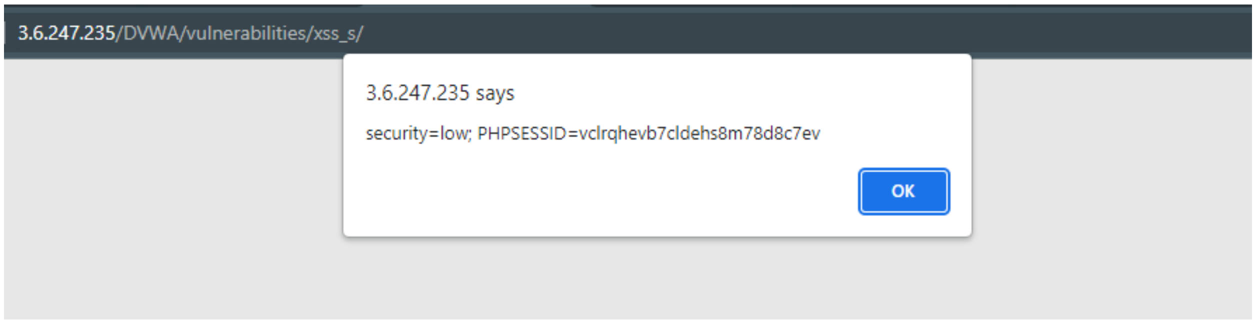
Name * hi

Message * <script>alert(document.cookie)</script>

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- 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/

Response

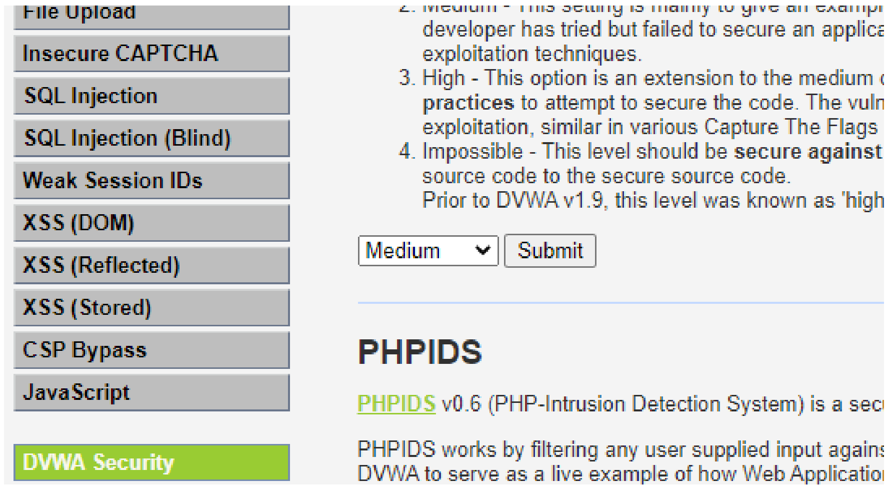

3.6.247.235/DVWA/vulnerabilities/xss_s/

3.6.247.235 says

security=low; PHPSESSID=vclrqhevb7cldehs8m78d8c7ev

This is stored XSS attack as the session id is persistent even after you change tabs and come back to this page. Every time we visit this page same popup is displayed. To clear these details and prevent the popup from re-appearing we need to clear the guestbook.

2. Does your attack work in "Medium" security level?



developer has tried but failed to secure an applica
exploitation techniques.
3. High - This option is an extension to the medium
practices to attempt to secure the code. The vuln
exploitation, similar in various Capture The Flags
4. Impossible - This level should be secure against
source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high

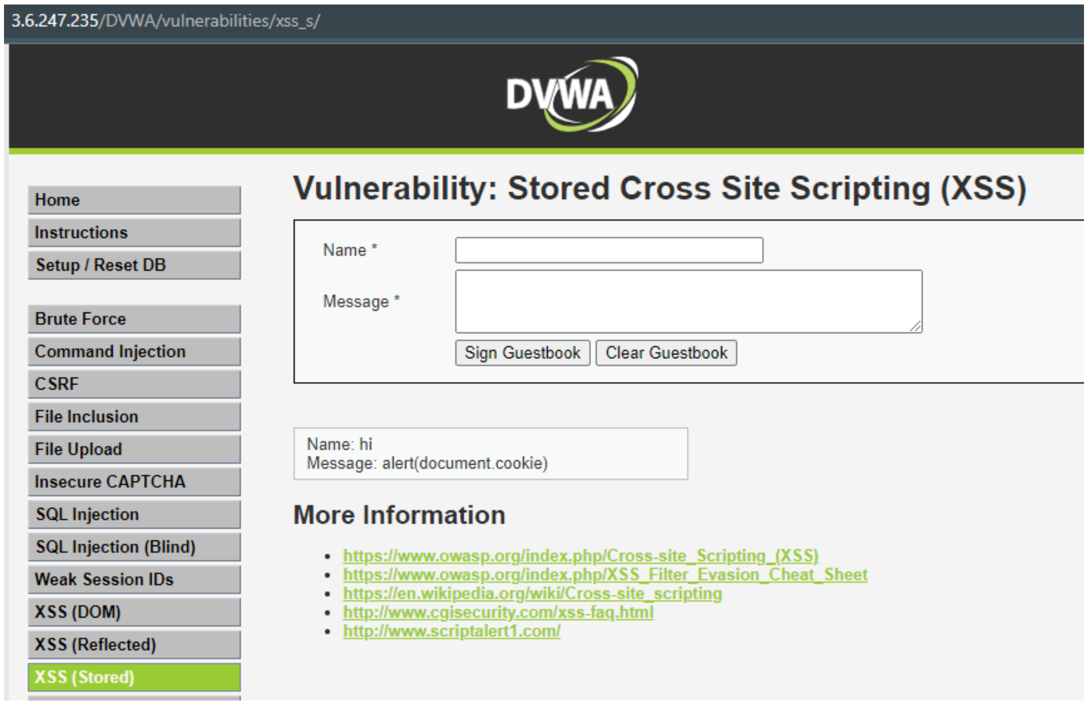
Medium Submit

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a sec

PHPIDS works by filtering any user supplied input agains
DVWA to serve as a live example of how Web Applicatio

The same attack as in question 1 did not work but we now modified the length of **Name** field via UI and the attack worked. Let us look at it in more detail below:
In medium security level entering the same input (**<script>alert(document.cookie)</script>**) as demonstrated in question 1 does not render the session id. Upon inspecting the page, it appears that the input was sanitized and rendered on UI. <script> tags were stripped which prevented any js insertion via message.


3.6.247.235/DVWA/vulnerabilities/xss_s/

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

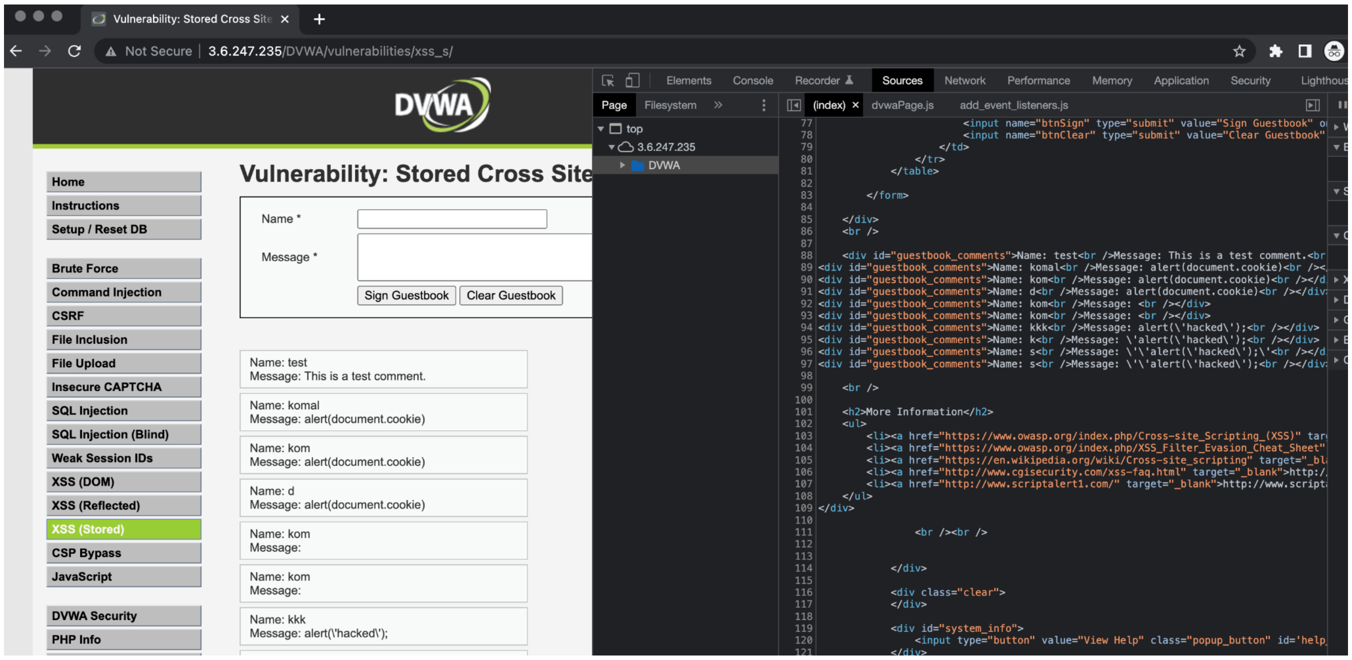
Name: hi
Message: alert(document.cookie)

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- 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/

We tried to modify the message as follows:
<sc<script>ript>alert('hacked');</script>

But this again did not invoke any alert. Also we could see in the page that 'hacked' keyword was escaped. This meant that the output was escaped.


Vulnerability: Stored Cross Site

Not Secure | 3.6.247.235/DVWA/vulnerabilities/xss_s/

Vulnerability: Stored Cross Site

Name *

Message *

Name: test
Message: This is a test comment.

Name: komal
Message: alert(document.cookie)

Name: kom
Message: alert(document.cookie)

Name: d
Message: alert(document.cookie)

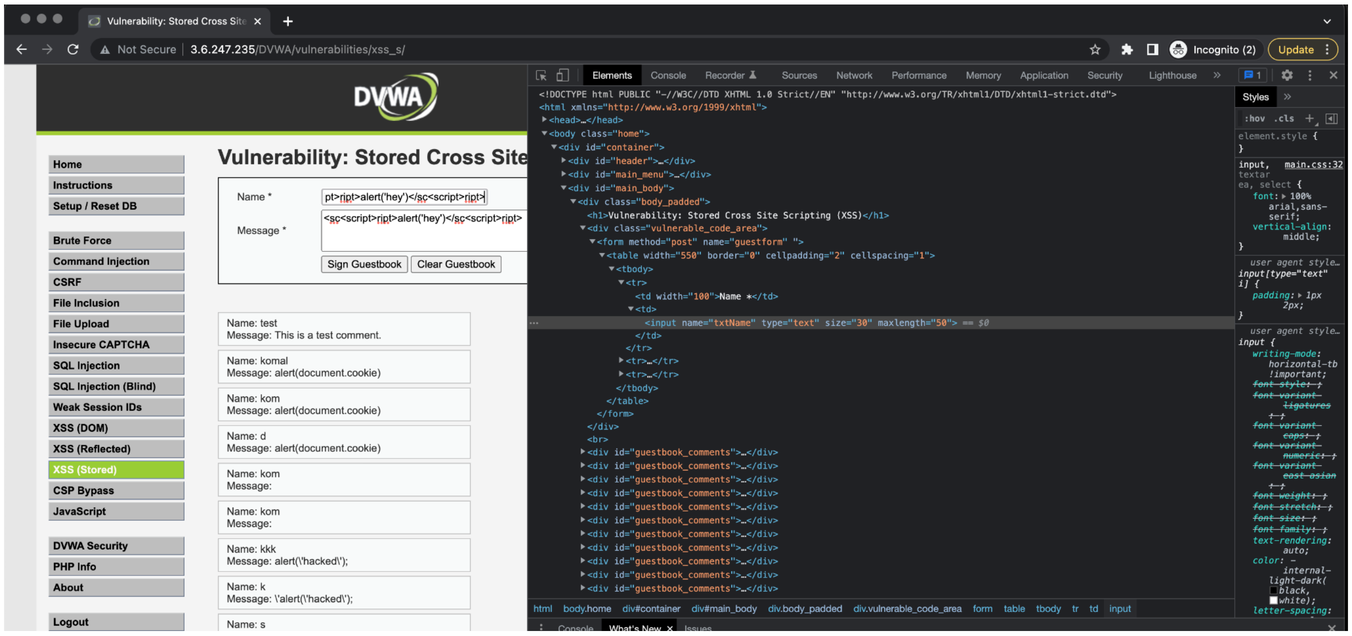
Name: kom
Message:

Name: kom
Message:

Name: kkk
Message: alert(\'hacked\');

```
                        <input name="btnSign" type="submit" value="Sign Guestbook" o
                        <input name="btnClear" type="submit" value="Clear Guestbook"
                    </td>
                </tr>
            </table>

        </form>

    </div>
    <br />

    <div id="guestbook_comments">Name: test<br />Message: This is a test comment.<br
<div id="guestbook_comments">Name: komal<br />Message: alert(document.cookie)<br /><
<div id="guestbook_comments">Name: kom<br />Message: alert(document.cookie)<br /></d
<div id="guestbook_comments">Name: d<br />Message: alert(document.cookie)<br /></div
<div id="guestbook_comments">Name: kom<br />Message: <br /></div>
<div id="guestbook_comments">Name: kom<br />Message: <br /></div>
<div id="guestbook_comments">Name: kkk<br />Message: alert(\'hacked\');<br /></div>
<div id="guestbook_comments">Name: k<br />Message: \'alert(\'hacked\');<br /></div>
<div id="guestbook_comments">Name: s<br />Message: \'\'alert(\'hacked\');\'<br /></d
<div id="guestbook_comments">Name: s<br />Message: \'\'alert(\'hacked\');<br /></div

    <br />

    <h2>More Information</h2>
    <ul>
        <li><a href="https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)" tar
        <li><a href="https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet"
        <li><a href="https://en.wikipedia.org/wiki/Cross-site_scripting" target="_bl
        <li><a href="http://www.cgisecurity.com/xss-faq.html" target="_blank">http:/
        <li><a href="http://www.scriptalert1.com/" target="_blank">http://www.script
    </ul>
</div>

                <br /><br />


            </div>

            <div class="clear">
            </div>

            <div id="system_info">
                <input type="button" value="View Help" class="popup_button" id='help
            </div>
```

However, now we updated the length of Name field and sent following script in name and we could see the alert.



3.6.247.235/DVWA/vulnerabilities/xss_s/

Vulnerability: Stored Cross Site

Name * pt>ript>alert('hey')</sc<script>ript>

Message * <sc<script>ript>alert('hey')</sc<script>ript>

Name: test
Message: This is a test comment.

Name: komal
Message: alert(document.cookie)

Name: kom
Message: alert(document.cookie)

Name: d
Message: alert(document.cookie)

Name: kom
Message:

Name: kom
Message:

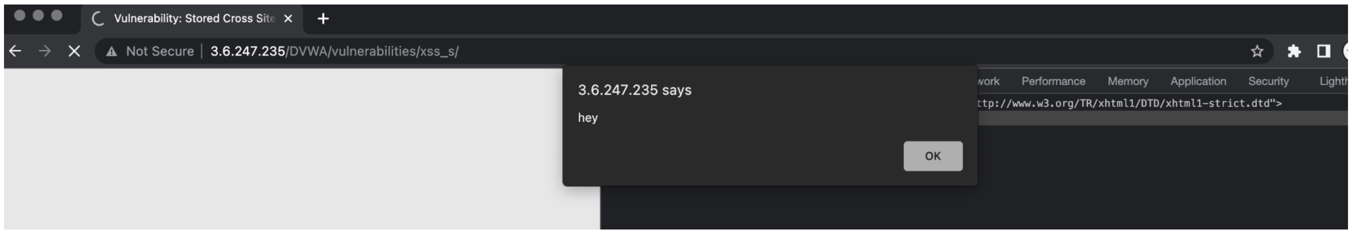
Name: kkk
Message: alert(\'hacked\');

Name: k
Message: \'alert(\'hacked\');

Name: s

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 ▶<head>…</head>
 ▼<body class="home">
   ▼<div id="container">
     ▶<div id="header">…</div>
     ▶<div id="main_menu">…</div>
     ▼<div id="main_body">
       ▼<div class="body_padded">
          <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>
         ▼<div class="vulnerable_code_area">
           ▼<form method="post" name="guestform" ">
             ▼<table width="550" border="0" cellpadding="2" cellspacing="1">
               ▼<tbody>
                 ▼<tr>
                    <td width="100">Name *</td>
                   ▼<td>
                      <input name="txtName" type="text" size="30" maxlength="50"> == $0
                    </td>
                  </tr>
                 ▶<tr>…</tr>
                 ▶<tr>…</tr>
                </tbody>
              </table>
            </form>
          </div>
          <br>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
```

html body.home div#container div#main_body div.body_padded div.vulnerable_code_area form table tbody tr td input



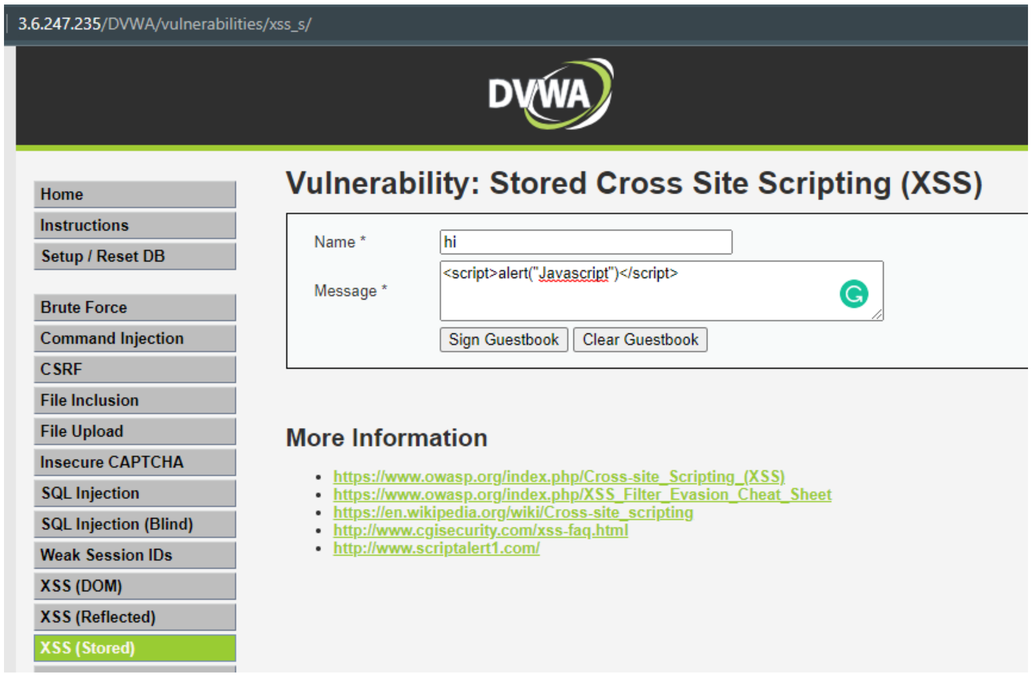
3.6.247.235/DVWA/vulnerabilities/xss_s/

3.6.247.235 says

hey

This shows that **Name field** does not have length validation or sanitization check on the backend. We could run the simple <script>alert('hey')</script> query on name field and it was still giving the alert box.

3. **Set the security mode to "Low" and examine the code that is vulnerable, and then set the security mode to "High" and reexamine the same code. What changed? How do the changes prevent attack from succeeding?**
   In low security level <script>alert("Javascript")</script> renders alert message on the browser as below:


3.6.247.235/DVWA/vulnerabilities/xss_s/

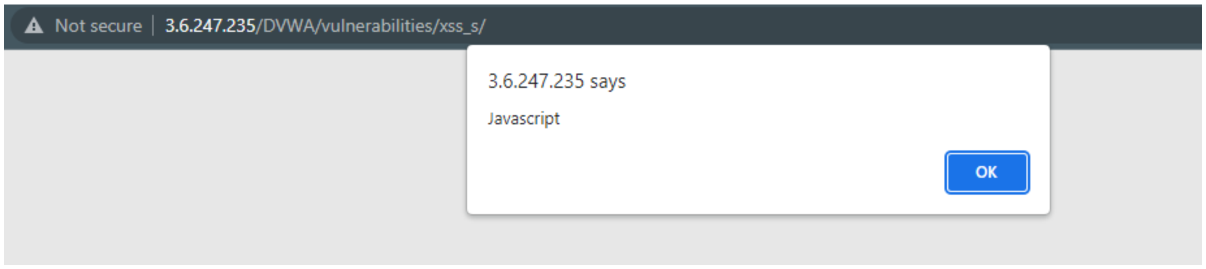
Vulnerability: Stored Cross Site Scripting (XSS)

Name * hi

Message * <script>alert("Javascript")</script>

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- 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/

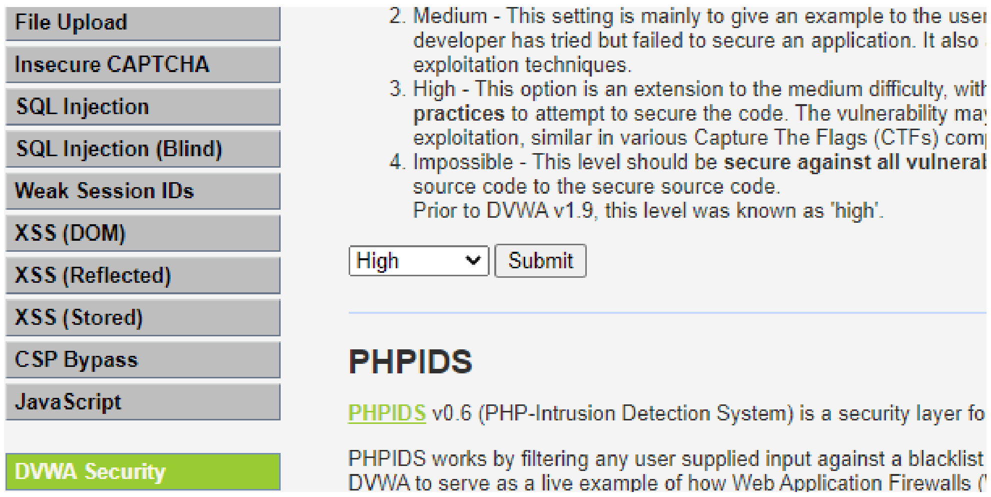

Not secure | 3.6.247.235/DVWA/vulnerabilities/xss_s/

3.6.247.235 says

Javascript

On refreshing the page, same alert message gets displayed because XSS playload is stored in the GuestBook.

Setting the security level to High:



2. Medium - This setting is mainly to give an example to the user
developer has tried but failed to secure an application. It also
exploitation techniques.
3. High - This option is an extension to the medium difficulty, with
practices to attempt to secure the code. The vulnerability may
exploitation, similar in various Capture The Flags (CTFs) com
4. Impossible - This level should be **secure against all vulneral**
source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

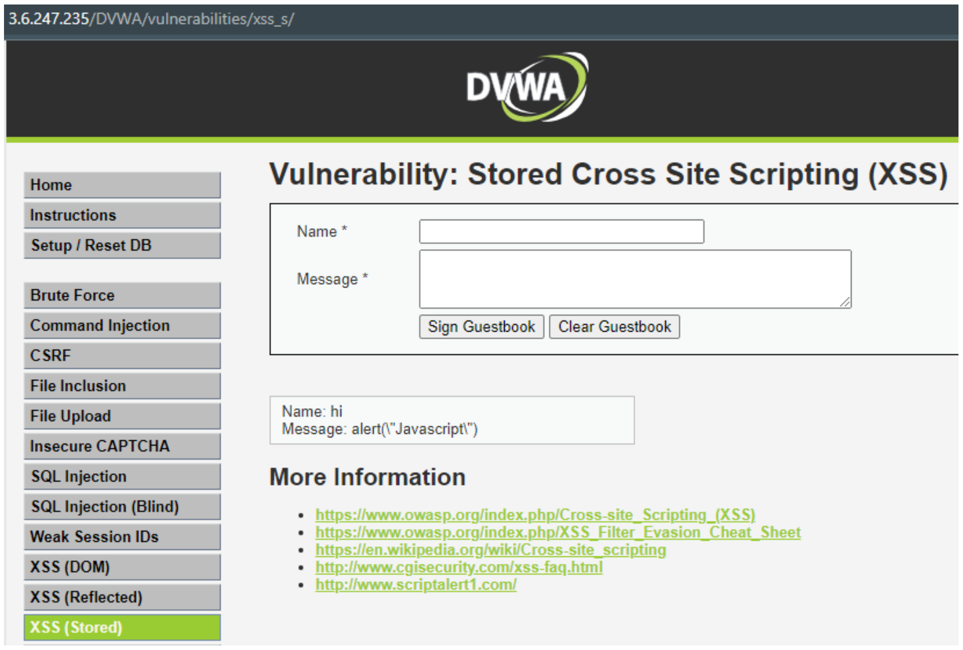
High Submit

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer fo

PHPIDS works by filtering any user supplied input against a blacklist
DVWA to serve as a live example of how Web Application Firewalls (

Entering the same input does not render alert message.



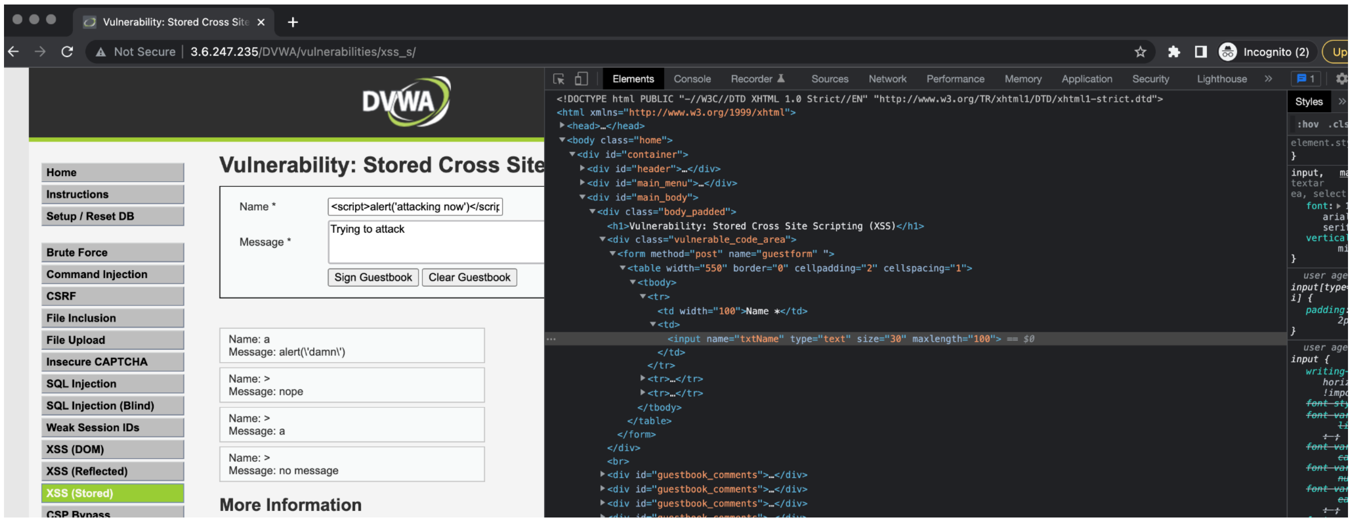
Vulnerability: Stored Cross Site Scripting (XSS)

Name: hi
Message: alert(\"Javascript\")

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- 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/

There may be chances that the backend code is performing input sanitization or html encoding on message field when it is accepting user's input. Looking at the server code it is clear that the message field is using two levels of sanitization. Firstly, **strip_tags()** is being used. This function removes all the HTML tags from the message field and even if some text contains quotes or bad character passes through this function, **htmlspecialchars()** will definitely encode into equivalent html character. So XSS payload becomes useless when the security level is set to high. Hence, message field is completely secure, and we cannot inject any XSS payload into it. Also, above screenshot shows that the strip_tags() function has removed the script tag from the message and we see only alert() text in the response. Also, there seem to be a code that prevents any occurrence of <script > tag. For instance,
<sc<script>ript>alert('hey')</script>
If only <script> is removed, the output will still be:
<script>alert('hey')</script>
But all the occurrences of <script> tags are getting stripped off while getting rendered on UI. We saw in question 2 that name field was rendering the alerts but it is not happening in High mode because the name field has strip_tags validations.



Vulnerability: Stored Cross Site

Not Secure | 3.6.247.235/DVWA/vulnerabilities/xss_s/

Name * `<script>alert('attacking now')</scrip`

Message * Trying to attack

Sign Guestbook | Clear Guestbook

Name: a
Message: alert(\'damn\')

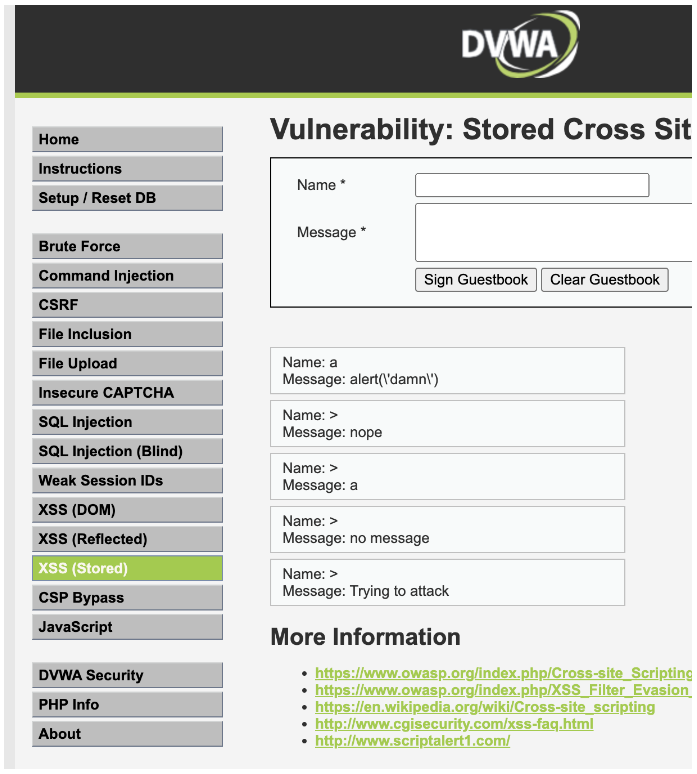
Name: >
Message: nope

Name: >
Message: a

Name: >
Message: no message

More Information

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 ▶<head>…</head>
 ▼<body class="home">
   ▼<div id="container">
     ▶<div id="header">…</div>
     ▶<div id="main_menu">…</div>
     ▼<div id="main_body">
       ▼<div class="body_padded">
          <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>
         ▼<div class="vulnerable_code_area">
           ▼<form method="post" name="guestform" ">
             ▼<table width="550" border="0" cellpadding="2" cellspacing="1">
               ▼<tbody>
                 ▼<tr>
                    <td width="100">Name *</td>
                   ▼<td>
                      <input name="txtName" type="text" size="30" maxlength="100"> == $0
                    </td>
                  </tr>
                 ▶<tr>…</tr>
                 ▶<tr>…</tr>
                </tbody>
              </table>
            </form>
          </div>
          <br>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
         ▶<div id="guestbook_comments">…</div>
```



Vulnerability: Stored Cross Sit

Name *

Message *

Sign Guestbook | Clear Guestbook

Name: a
Message: alert(\'damn\')

Name: >
Message: nope

Name: >
Message: a

Name: >
Message: no message

Name: >
Message: Trying to attack

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting
- https://www.owasp.org/index.php/XSS_Filter_Evasion
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/