

HTML Injection

Another major aspect of front end security is validating and sanitizing accepted user input. In many cases, user input validation and sanitization is carried out on the back end. However, some user input would never make it to the back end in some cases and is completely processed and rendered on the front end. Therefore, it is critical to validate and sanitize user input on both the front end and the back end.

HTML injection occurs when unfiltered user input is displayed on the page. This can either be through retrieving previously submitted code, like retrieving a user comment from the back end database, or by directly displaying unfiltered user input through **JavaScript** on the front end.

When a user has complete control of how their input will be displayed, they can submit **HTML** code, and the browser may display it as part of the page. This may include a malicious **HTML** code, like an external login form, which can be used to trick users into logging in while actually sending their login credentials to a malicious server to be collected for other attacks.

Another example of **HTML Injection** is web page defacing. This consists of injecting new **HTML** code to change the web page's appearance, inserting malicious ads, or even completely changing the page. This type of attack can result in severe reputational damage to the company hosting the web application.

Example

The following example is a very basic web page with a single button "**Click to enter your name**." When we click on the button, it prompts us to input our name and then displays our name as "**Your name is ...**":



If no input sanitization is in place, this is potentially an easy target for **HTML Injection** and **Cross-Site Scripting (XSS)** attacks. We take a look at the page source code and see no input sanitization in place whatsoever, as the page takes user input and directly displays it:

Code: **html**

```
<!DOCTYPE html>
<html>

<body>
  <button onclick="inputFunction()">Click to enter your name</button>
  <p id="output"></p>

  <script>
    function inputFunction() {
      var input = prompt("Please enter your name", "");

      if (input != null) {
        document.getElementById("output").innerHTML = "Your name is " + input;
      }
    }
  </script>
</body>

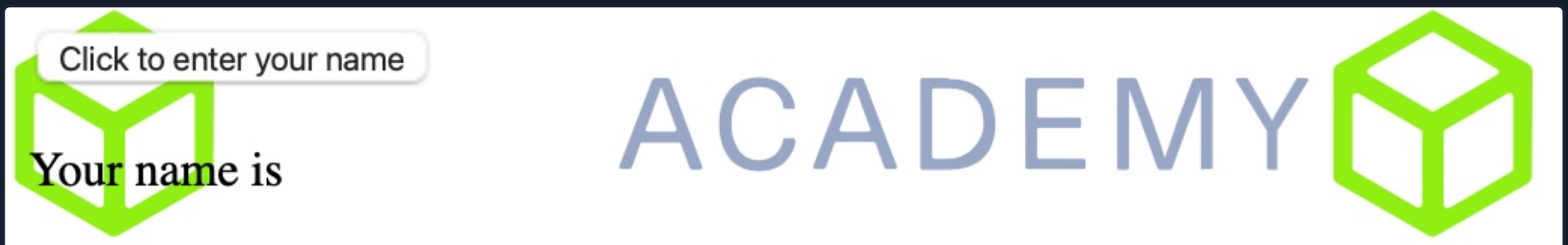
</html>
```

To test for **HTML Injection**, we can simply input a small snippet of **HTML** code as our name, and see if it is displayed as part of the page. We will test the following code, which changes the background image of the web page:

Code: **html**

```
<style> body { background-image: url('https://academy.hackthebox.com/images/logo.svg'); } </style>
```

Once we input it, we see that the web page's background image changes instantly:



In this example, as everything is being carried out on the front end, refreshing the web page would reset everything back to normal.

Start Instance

1 / 1 spawns left

Waiting to start...

Questions

Answer the question(s) below to complete this Section and earn cubes!

Target: [Click here to spawn the target system!](#)


+ 1 


What text would be displayed on the page if we use the following payload as our input: Click Me

Submit your answer here...

 Submit

 Hint

 Previous

Next 








 Go to Questions

Table of Contents





Introduction to Web Applications

Introduction	
Web Application Layout	
Front End vs. Back End	


Front End Components

HTML	
Cascading Style Sheets (CSS)	
JavaScript	

Front End Vulnerabilities

 Sensitive Data Exposure	
 HTML Injection	
 Cross-Site Scripting (XSS)	
Cross-Site Request Forgery (CSRF)	

Back End Components

Back End Servers	
Web Servers	
Databases	
 Development Frameworks & APIs	

Back End Vulnerabilities

Common Web Vulnerabilities

Public Vulnerabilities

Next Steps

Next Steps

My Workstation

OFFLINE

Start Instance

1 / 1 spawns left