SCENARIO

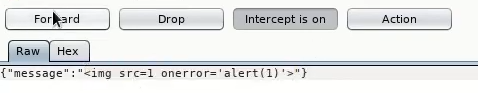
The application has a web-socket implemented live chat feature which are viewed by a support agent in real time but there exists a vulnerability of injecting JavaScript into it which will trigger an alert on the support agent’s browser.

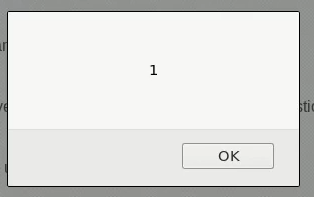
**PROCEDURE**

1. Go to Live Chat option.
2. Start BurpSuite’s Interceptor.
3. Then type and send any special character into the chat box and we can see that the browser HTML encodes the message before sending.
4. Now, send another random text message and using BurpSuite’s Proxy’s Intercept modify the message with the given payload.
5. After injecting the payload there will be an alert on your side as well as the client’s side.

# PAYLOAD

<img src=1 onerror='alert(1)'>

**PROOF OF CONCEPT**

****

**REMEDIATIONS**

1. **Validate and Sanitize Input:** Every piece of data that is transmitted over the websockets should be treated as untrusted, regardless of the source. Always validate and sanitize user input before processing. Libraries like DOMPurify can be used to sanitize content.
2. **Websocket Filtering:** Just like how web traffic can be filtered for malicious content, websocket traffic should also be monitored and filtered. Look for suspicious payloads and patterns, and drop or sanitize them.
3. **Use Websocket Libraries Wisely:** If using third-party libraries to implement websockets, ensure that they are well-maintained, have a good reputation, and are regularly updated. It's also crucial to keep them up-to-date, as patches for known vulnerabilities are released regularly.
4. **Escape Data:** Instead of injecting user data directly into the chat box, HTML entities in the data should be escaped. This ensures that any code in the data is treated as string data and not executable code.
5. **Content-Type Headers:** Ensure that the correct Content-Type header is being set for data that's being returned. This helps prevent certain types of XSS attacks where the browser misinterprets the data type.