SCENARIO

The application has an aggressive but flawed XSS filter in the live chat functionality as it blocks an IP if it detects any JavaScript code being injected at a wrong place. We will try to exploit the vulnerability to generate an alert on the client’s browser.

# PROCEDURE

1. Go to live chat option.
2. Start BurpSuite’s Interceptor.
3. Then send a message to the support agent, then we will go to the BurpSuite’s Proxy’s WebSocket History and send that request to our BurpSuite’s Repeater.
4. We noticed that if we replay the same request with the payload then our IP gets blocked.
5. So, we will bypass that by using X-Forwarded-For header in order to spoof our IP address.
6. Then we will send the request again and it will be successful.

# PAYLOAD

1. <img src=1 oNeRrOr=alert`1`>
2. X-Forwarded-For: 1.1.1.1

# REMEDIATION

1. **Improved Input Sanitization:** The application should sanitize all inputs before processing them. Rather than looking for specific patterns of malicious input (which can often be bypassed with clever encoding or other techniques), the application should whitelist allowed input and reject anything that doesn't match. For example, if a field is expected to contain only alphanumeric characters, any input containing special characters should be rejected.
2. **Content Security Policy (CSP):** Implement a strict CSP that will help prevent XSS attacks. This policy can be used to specify which script sources are allowed, thus preventing the execution of malicious inline scripts.
3. **X-Forwarded-For Limitation:** The application should not rely solely on the X-Forwarded-For header for IP filtering, as this header can be easily spoofed. Instead, the application should use a combination of techniques and checks to determine the originating IP address.
4. **Implement Rate Limiting:** Even if an attacker can spoof their IP, rate limiting can help mitigate the risk of brute-force attacks by limiting the number of requests from a particular IP or IP range in a given time frame.