SCENARIO

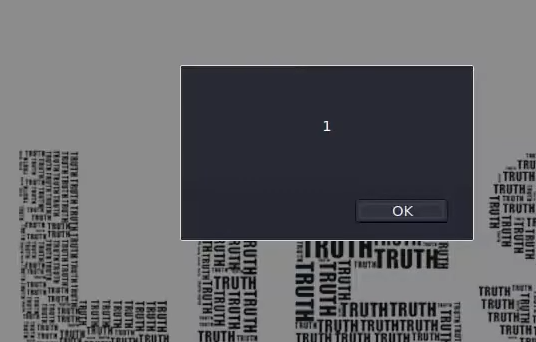
The application contains a simple stored cross site scripting vulnerability in the comment functionality which is shown to every user who views the injected blog so we will try to exploit it by injecting some payload containing malicious script in the comment box.

**PROCEDURE**

1. Go to the vulnerable web application.
2. Try to post any comment and study the request that is made to the server using BurpSuite’s Interceptor and we noticed that the request is sent using the URL.
3. So, we will modify the comment parameter with all the fields well filled except the comment in order to inject the payload to generate a basic alert on the page.
4. Now we get an alert, it is clear that we can exploit the application more by injecting reflected XSS.

**PAYOAD**

<script>alert("Hey!")</script>

**PROOF OF CONCEPT**

**REMEDIATION**

1. **Sanitize Input:** Implement a robust input sanitation mechanism to filter out any potentially malicious payloads from being saved. Libraries like OWASP’s Java Encoder for Java applications or DOMPurify for JavaScript can be utilized.
2. **Database Escaping:** Ensure that all inputs are escaped correctly before they are stored in the database. This will prevent any stored script from executing when fetched and displayed.
3. **Content Security Policy (CSP):** Implement a strong Content Security Policy to restrict the sources from which scripts can be executed. This can help in reducing the impact even if an attacker tries to exploit an XSS vulnerability.