SCENARIO

The application contains a DOM based open redirection vulnerability which redirects the user to our exploit server. We’ll try to exploit this vulnerability by forcing the user to redirect to the exploit server.

**PROCEDURE**

1. Access the application and open any blog to inspect its source code.
2. Upon inspecting we see that there is a button for going back to the home page which includes:

<a href='#' onclick='returnURL' = /url=https?:\/\/.+)/.exec(location); if(returnUrl)location.href = returnUrl[1];else location.href = "/"'>Back to Blog</a>

1. We notice that the URL parameter contains an open redirection vulnerability that allows you to change the location where the Back to blog button takes you.
2. So, according to that we will create a payload and navigate to it by injecting it into our URL which will change the URL parameter.

**PAYLOAD**

https://YOUR-LAB-ID.web-security-academy.net/post?postId=4&url=https://YOUR-EXPLOIT-SERVER-ID.exploit-server.net/

**PROOF OF CONCEPT**

****

**REMEDIATION**

1. **Strict Validation of URL Parameters:** Always validate and sanitize input parameters. In the case of URLs, only allow redirection to a predefined list of safe URLs or domains.
2. **Disallow Full URLs:** If your redirection logic doesn't require external domains, then only capture the path and/or the query parameters, rather than the full URL.
3. **Use a Token System:** Instead of allowing any redirection URL, assign specific tokens to specific URLs and only allow redirection using those tokens.
4. **Referrer Checking:** Before executing the redirect, ensure that the referrer is a trusted source or matches a list of trusted domains.
5. **Use Meta Refresh or JavaScript:** Instead of using a 302 redirect or altering location.href directly, use a meta refresh or a JavaScript prompt to inform users that they are leaving the site and give them a chance to approve or cancel the action.