SCENARIO

The application is vulnerable to server-side template injection due to the unsafe construction of an ERB template. We will try to figure out the way to execute arbitrary code on the backend server.

**PROCEDURE**

1. Open the application and click on the first product itself to get more information about it.
2. We see that the application renders a message saying that the product is out of stock.
3. Notice the URL and we see that the message parameter is appended into it.
4. Now, to identify the template we will try to execute the Payload 1 into the URL within the message parameter.
5. We see that instead of getting the supplied input reflected on the screen we get 49 as response which confirms that the server is using ERB template engine.
6. So, we will try to inject Payload 2 into the URL in the message parameter in order to do Remote Code Execution by going through the documentation of ERB template engine.

**PAYLOAD**

1. <%= 7\*7 %>
2. <%= system("rm /home/carlos/morale.txt") %>

**REMEDIATION**

1. **Avoid Dynamic Template Rendering:** Avoid allowing user input to dynamically select or influence templates. Any feature that allows users to control parts of a template can expose the application to server-side template injection attacks.
2. **Use Safe API Methods:** When constructing templates, always use methods or functions designed for safely creating templates. For instance, in Ruby on Rails, prefer the use of content\_tag helper over raw string concatenation.
3. **Input Validation and Sanitization:** Always validate and sanitize user inputs before they're processed. Ensure that the data being passed to templates is of the expected type and format. Reject any input that appears malicious or contains unexpected characters.
4. **Limit Template Features:** Disable unnecessary template features that can be used maliciously. For example, in ERB, you might limit the use of certain Ruby methods or objects within templates.
5. **Use Least Privilege:** The application should run with the least possible privileges. This can help in minimizing the damage in case an attacker manages to execute arbitrary commands.
6. **Escape User Input:** Always escape user input to ensure that it is treated as data and not executable code. This is especially important in contexts where user input is included in templates.