SCENARIO

The application is vulnerable to server-side template injection due to the way it unsafely uses a template. We will try to figure out a way to know the template used in order to execute arbitrary code on the backend server.

**PROCEDURE**

1. Open the application and log in using the provided credentials to get access to the content editing functionality.
2. Navigate to any blog and click the edit template button, we see that the application uses **${}** syntax to something invalid.
3. We see that the application throws an exception revealing the name of the template engine the server is configured with.
4. So, we will try to inject the Payload into the template in order to do Remote Code Execution by going through the documentation of Freemaker template engine.

**PAYLOAD**

<#assign ex="freemarker.template.utility.Execute"?new()> ${ ex("rm /home/carlos/morale.txt") }

**REMEDIATION**

1. **Avoid Direct Template Execution:** Always refrain from allowing user input to dictate or control template content directly. User input should never be processed directly as a template or be allowed to directly influence template structure.
2. **Safe API Methods:** Always prefer and use methods or functions designed to safely handle templates. If there's a safer way of using templates in Freemaker, always go for that.
3. **Sanitize and Validate Input:** Ensure that user inputs are sanitized and validated before they're processed. Confirm that user input passed into templates aligns with the expected type, structure, and format. Outrightly reject any input that contains unexpected characters or patterns.
4. **Disable Unneeded Template Features:** If certain template engine features aren't necessary for your application, disable them. By restricting the capabilities of the template engine, you can reduce the chances of exploitation.
5. **Use Least Privilege Principle:** Always ensure your application runs with the minimum required privileges. If the application doesn't require the ability to execute system commands, don't grant that privilege. This ensures that even if an attacker gains some form of execution, their capabilities are limited.
6. **Input Escaping:** Always escape user input so that it's interpreted as data and not executed as code. Especially in scenarios where user input is integrated into templates, it's important to ensure that the input is treated strictly as data.