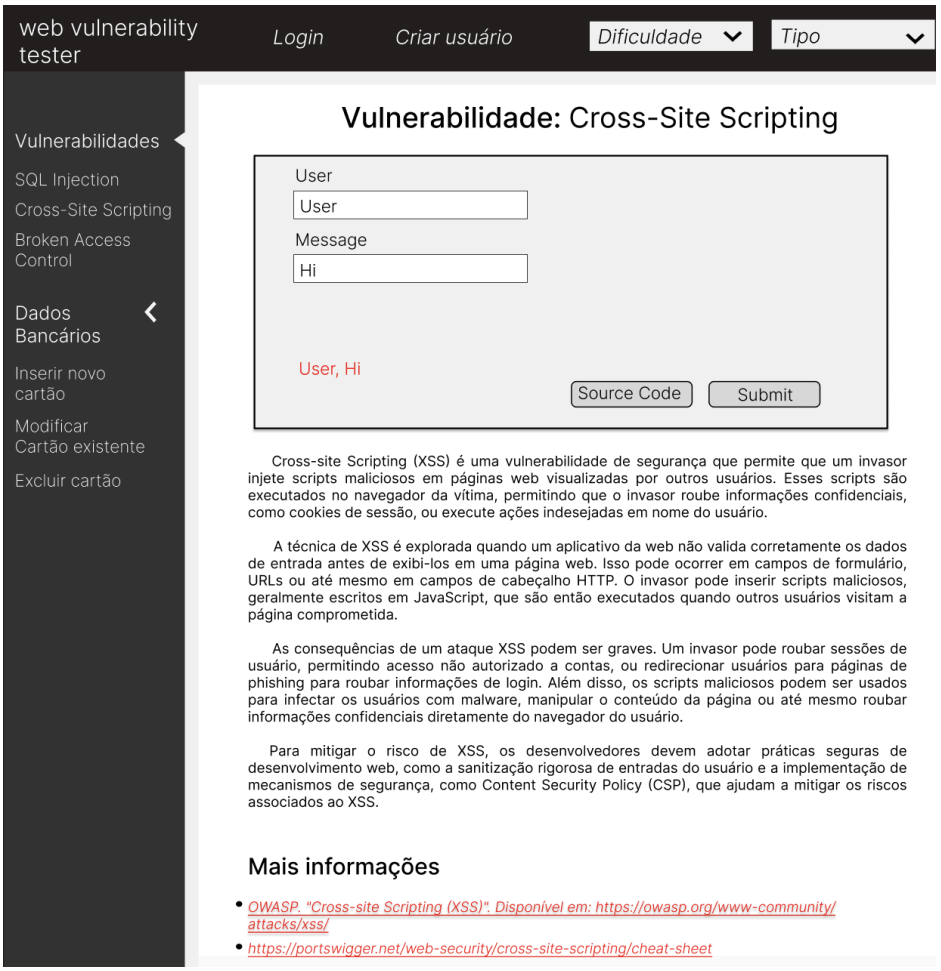
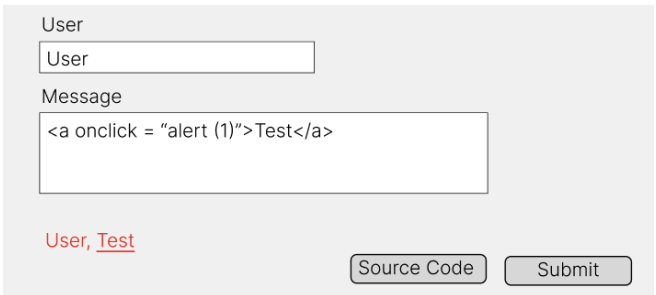
Tela Principal:



Redirecionar o usuário:

*Usuário pode executar os testes

Caixa de texto Cross-site:



Source code:

Código vulnerável

```
// Vulnerable code that directly inserts user input into the HTML content
function displayMessage(message) {
    // Display the message on the page
    document.getElementById('message-container').innerHTML = message;
}
```

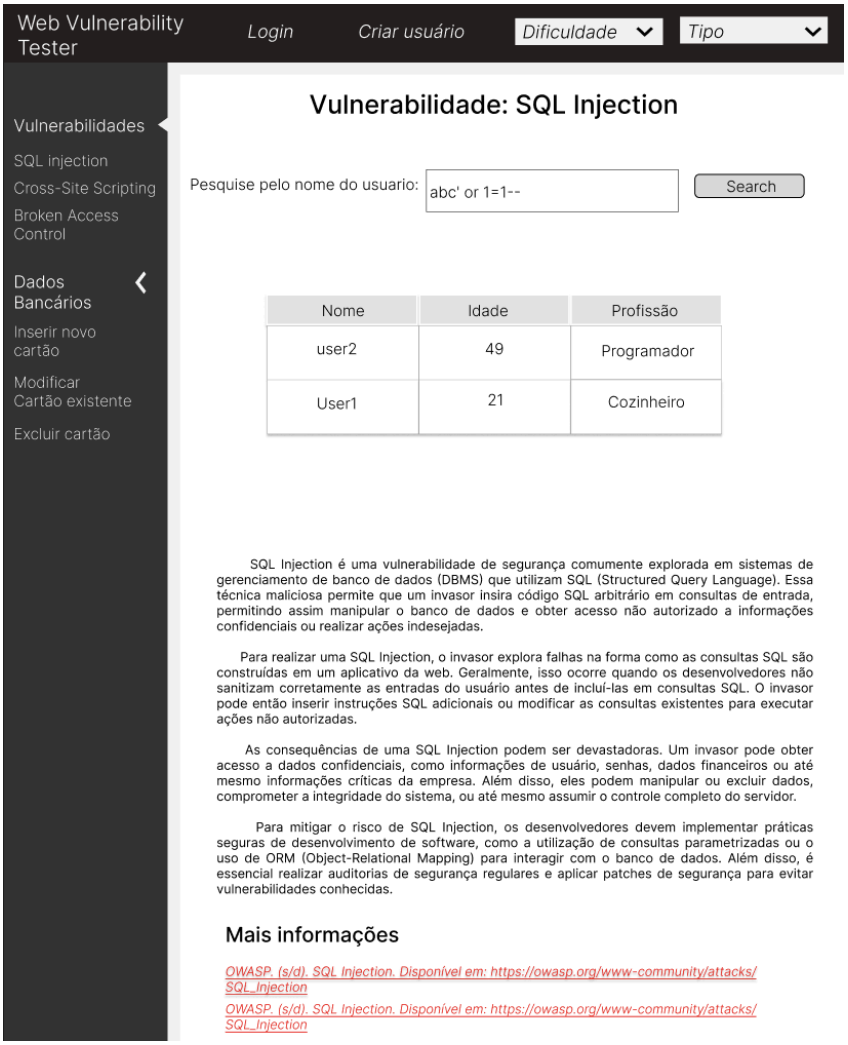
Código protegido

```
// Corrected code that properly escapes user input to prevent XSS attacks
function displayMessage(message) {
    // Escape HTML characters in the message to prevent XSS attacks
    var sanitizedMessage = escapeHTML(message);
    // Display the sanitized message on the page
    document.getElementById('message-container').textContent = sanitizedMessage;
}

// Function to escape HTML characters in a string
function escapeHTML(html) {
    return html.replace(/&/g, "&amp;").replace(/</g, "&lt;").replace(/>/g, "&gt;").re
}
```

*Mostra o código fonte do formulário vulnerável (por exemplo) e como protegê-lo

Outra página:



Pop-up com exemplo SQLi:

```
id=-1 UNION SELECT 1,table_name FROM information_schema.tables WHERE
table_type=0x6261736520746162c65 AND table_schema=0x64767761; &Submit=Submit
```

Dificuldade:

Fácil: campo livre para qualquer tipo de dado de submissão.

Médio: Campo do tipo select, que obriga o usuário a usar o próprio inspect da página para alterar os parâmetros de submissão. Além disso, alguns dados podem estar criptografadas com hash e afins. Talvez checar por caracteres xss.

Difícil: addslashes()" together with "mysql_real_escape_string()"

Outra página: