Лабораторная работа 8. Часть 2.

Предотвращение атак, связанных с XSS

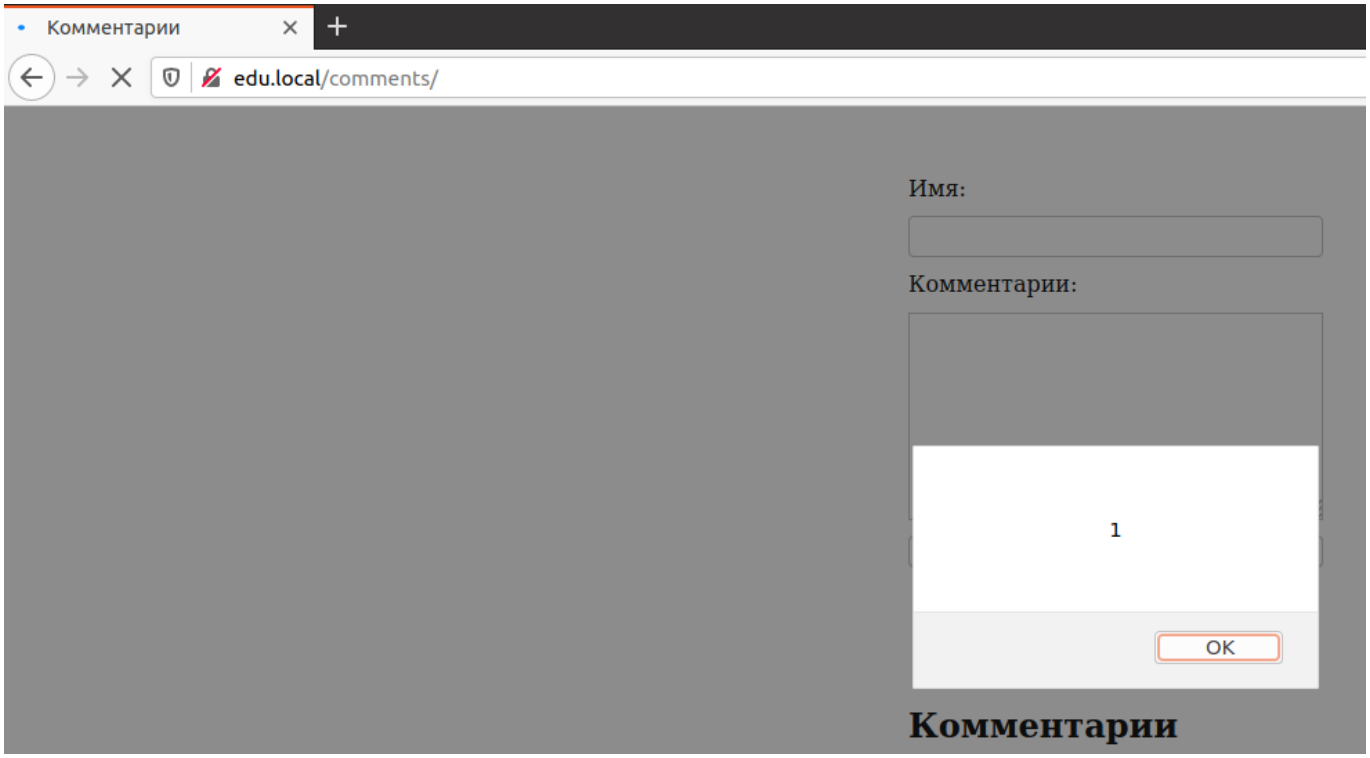
Индивидуальность отчетов:

Как минимум, в имени пользователя ОС

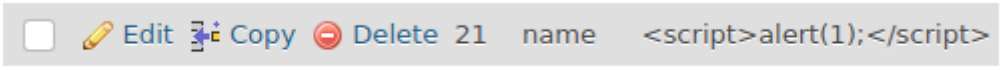
Обязательно скриншотить каждый этап и пояснять

Обязательно выводы должны быть

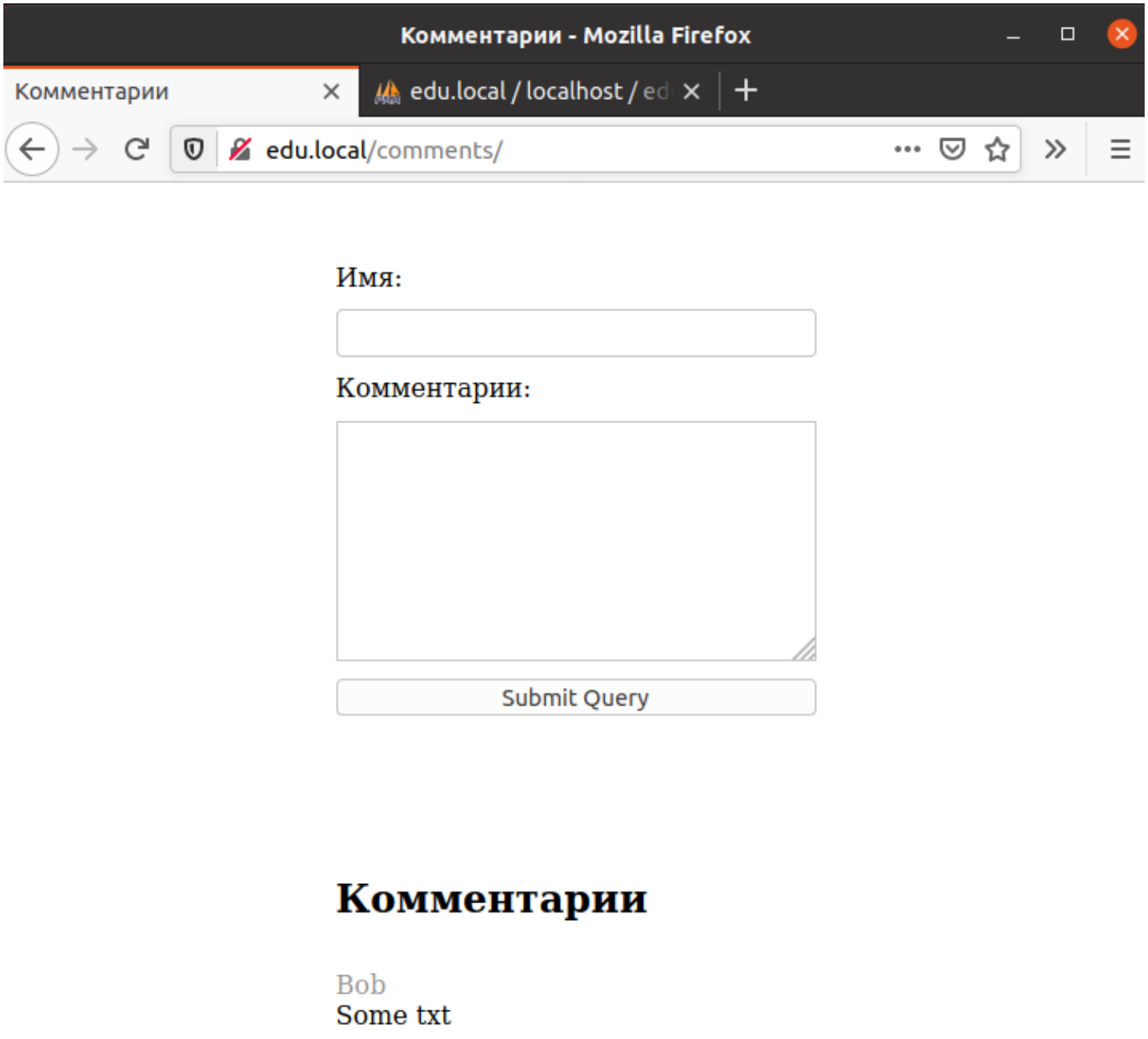
1. Убедиться в наличии уязвимости из части 1



2. Удалить внедренный JS-код из базы данных



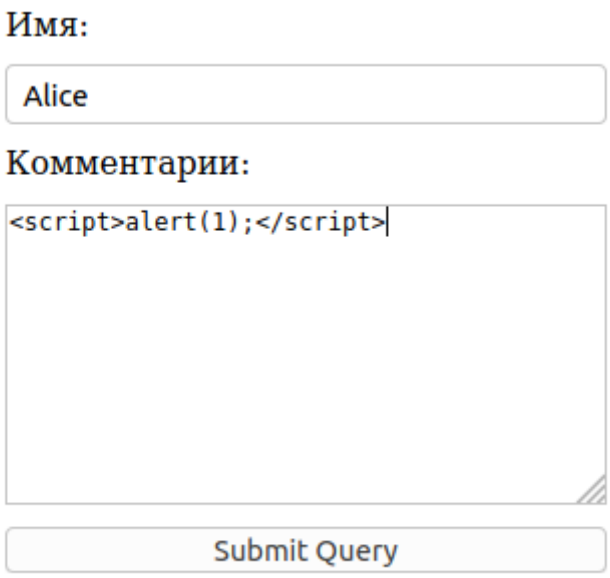
3. Убедиться, что внедренного кода больше нет



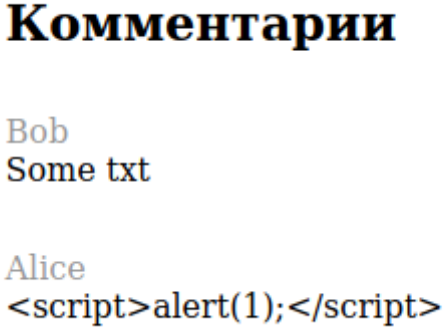
4. С помощью функции htmlspecialchars() закодировать переменные входных данных $name и $comment

```php
<?php
$dbh = new PDO('mysql:host=localhost;dbname=edu', 'newuser', '123');
// $name = $_POST["name"];
// $comment = $_POST["comment"];
$name = htmlspecialchars($_POST["name"]);
$comment = htmlspecialchars($_POST["comment"]);
```

5. Попробовать ещё раз внедрить вредоносный код



6. Убедиться, что мы защищены, не увидев результат выполнения скрипта



7. Посмотреть, как выглядит исходный код страницы и запись в базе данных

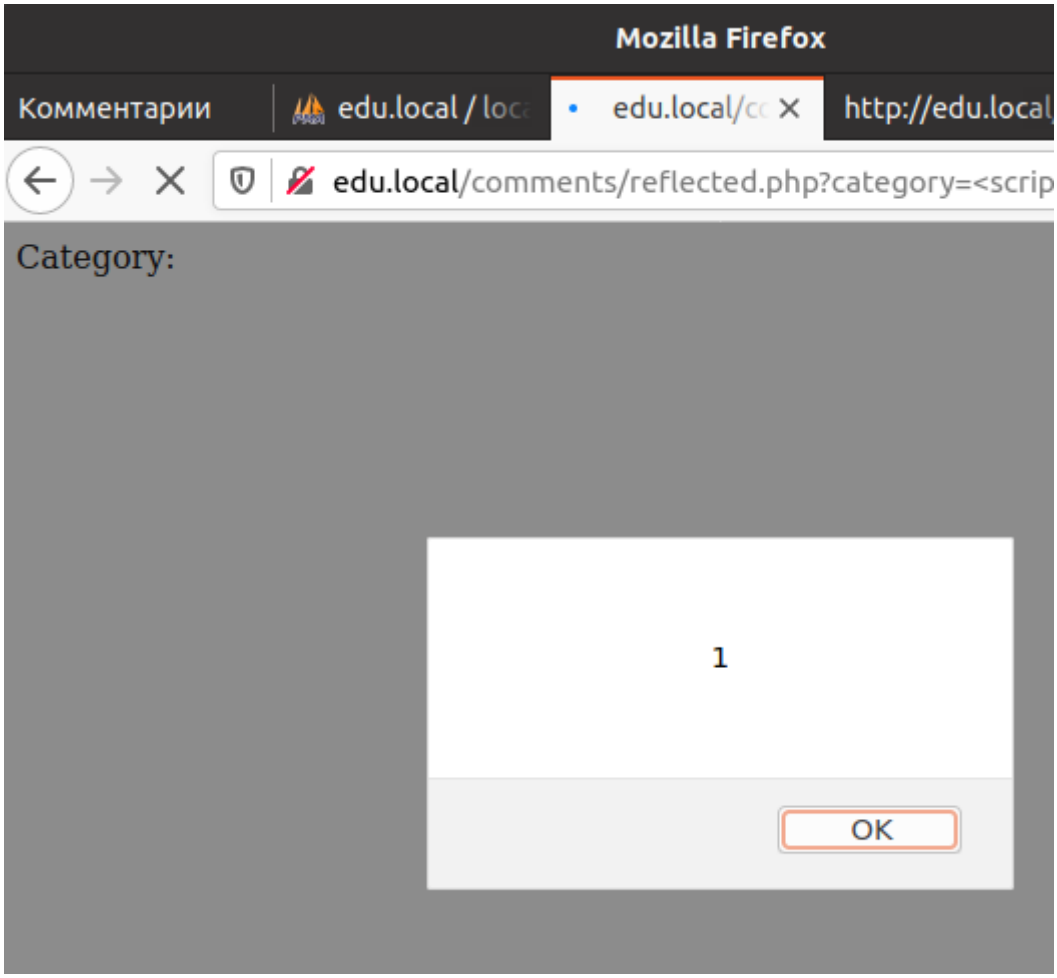
"> Alice <script>alert(1);</script>

Edit Copy Delete 24 Alice <script>alert(1);</script>

8. В отчете описать промежуточные выводы

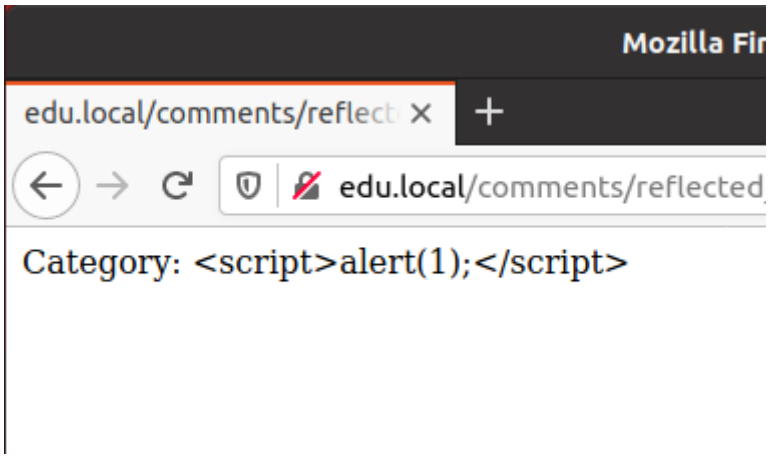
Уточнить, почему кодирования HTML достаточно в данном случае

9. По аналогии убедиться в том, что уязвимость существует и в reflected.php



10. Аналогичным методом защититься и посмотреть результат

На самой странице:



И в исходном коде:

```
1 <p>Category: &lt;script&gt;alert(1);&lt;/script&gt;</p>
```

11. В отчете описать промежуточные выводы

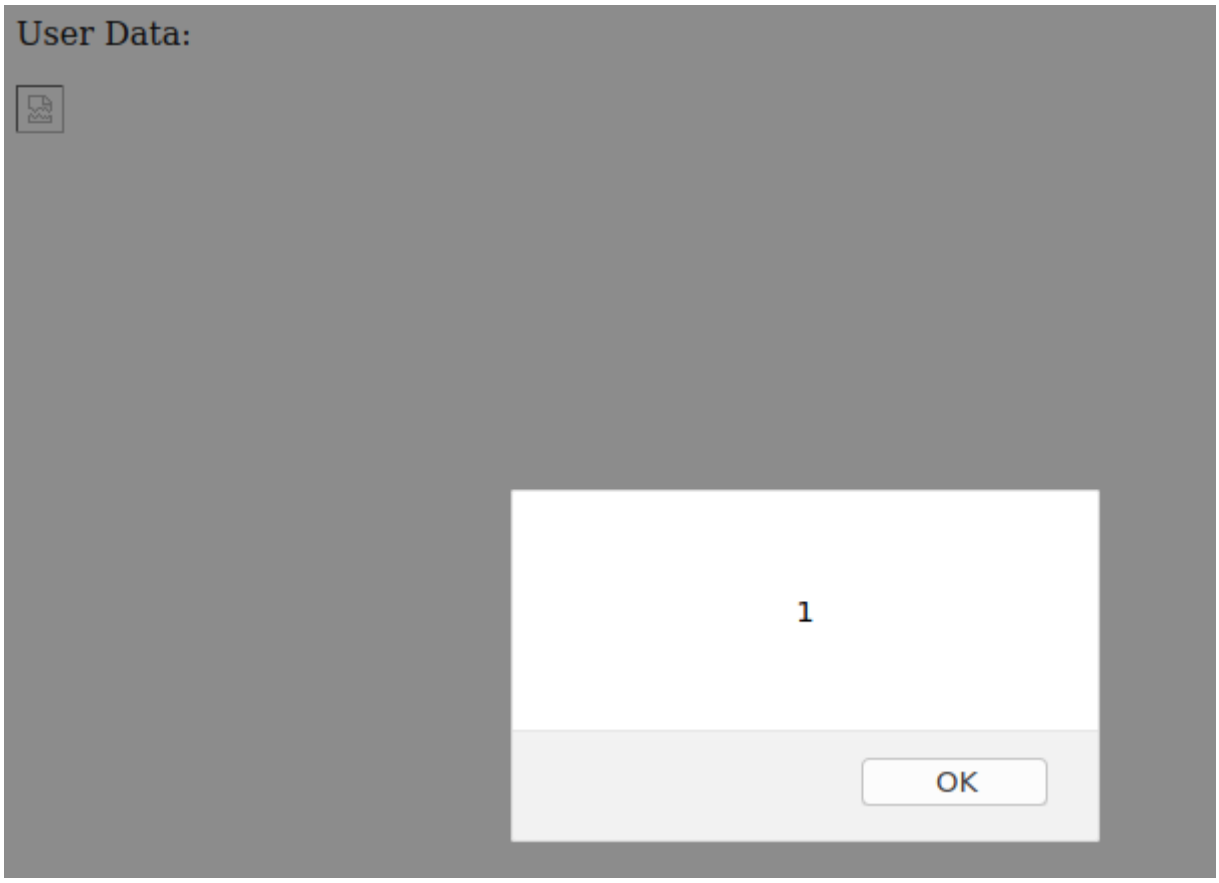
Уточнить, почему кодирования HTML достаточно в данном случае

12. Перед рассмотрением DOM-based XSS ответить на вопрос:

В какие функции JS в общем случае с точки зрения безопасности нельзя передавать пользовательские данные?

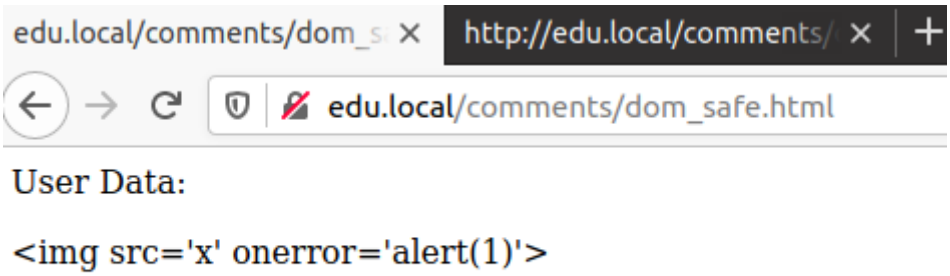
13. Рассмотреть следующий пример DOM-based XSS

```
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <p>User Data:</p>
    <p id="cid"></p>

    <script>
      const userData = "<img src='x' onerror='alert(1)'>";
      console.log(userData);
      document.getElementById("cid").innerHTML = userData;
    </script>
  </body>
</html>

```

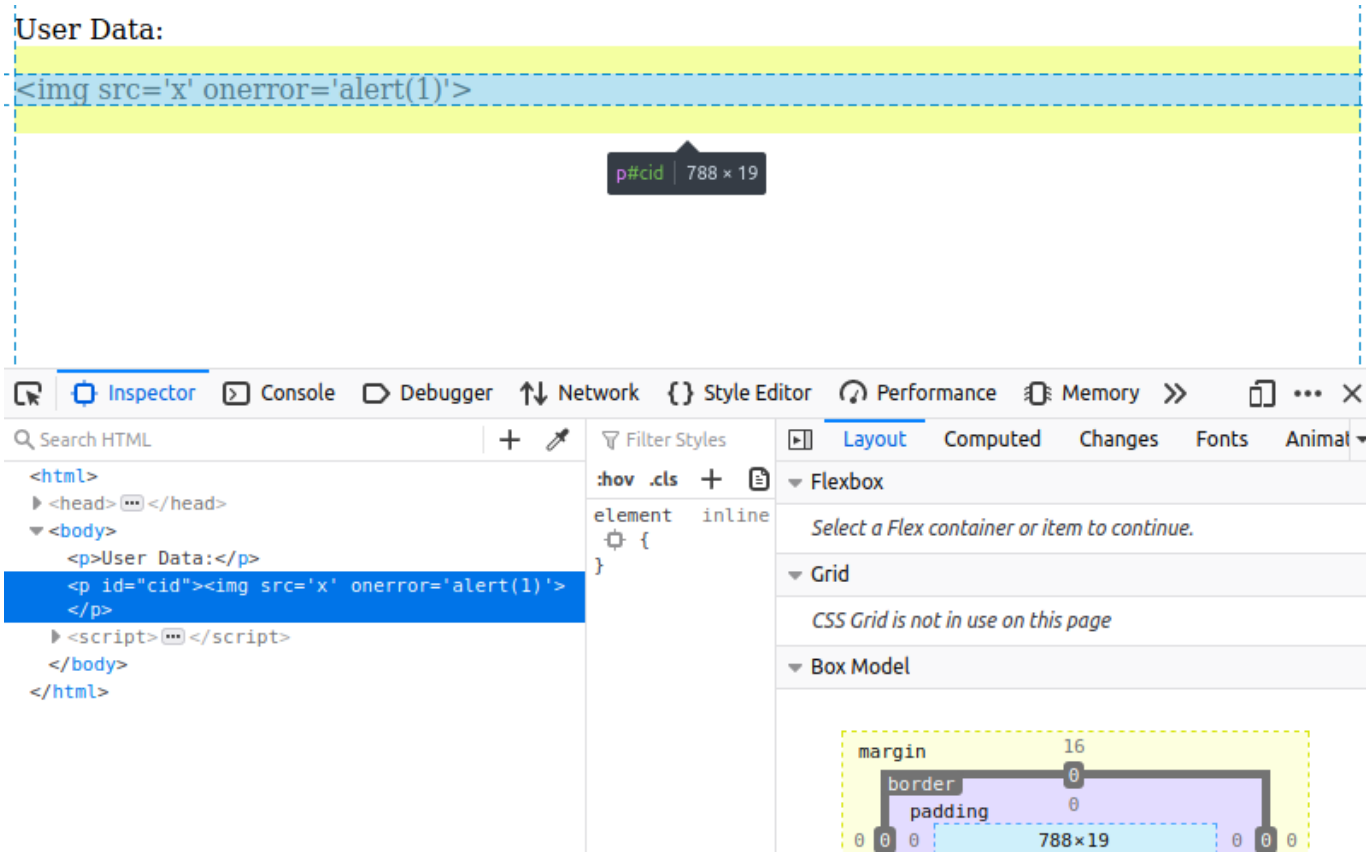


14. Изменить innerHTML на innerText и посмотреть на результат

На самой странице:



В инспекторе элементов:



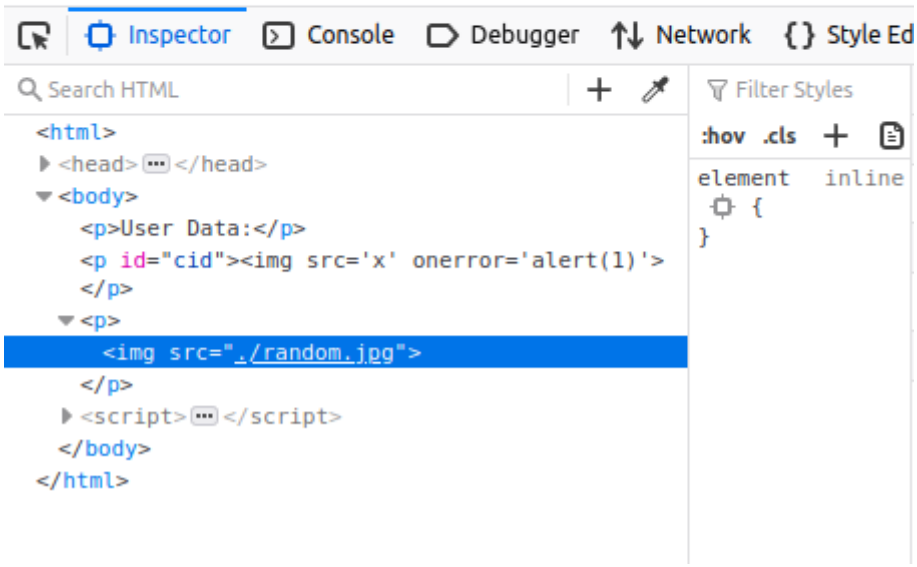
15. Сравнить различия с реальным тэгом <img>

Для этого добавим в HTML ещё одну картинку и ещё раз посмотрим на структуру дерева элементов

```
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <p>User Data:</p>
    <p id="cid"></p>
    <p><img src="./random.jpg"></p>

    <script>
      const userData = "<img src='x' onerror='alert(1)'>";
      console.log(userData);
      document.getElementById("cid").innerText = userData;
    </script>
  </body>
</html>

```

User Data:

<img src='x' onerror='alert(1)'>



В <p id=“cid”> нет внутри тэга <img>, внутри просто текст.

В <p>, следующем после него, вложен реальный тэг <img>.

11. В отчете описать промежуточные выводы

12. Сделать вывод по всей лабораторной работе