HxH404 / **OFOS-order-php-reflected-xss-cust_id.md** Secret

Last active 4 days ago

Code Revisions 2

Reflected XSS in code-projects.org Online Food Ordering System 1.0 - order.php cust_id parameter

`OFOS-order-php-reflected-xss-cust_id.md`

Reflected Cross-Site Scripting (XSS) in code-projects.org

Online Food Ordering System 1.0

The official source/reference of the product is: https://code-projects.org/online-food-ordering-system-in-php-with-source-code/

Vulnerability Details

Field	Value
Product	Online Food Ordering System
Version	1.0
Vendor	code-projects.org
Vulnerable File	`/form/order.php`
Parameter	`cust_id`
Request Method	GET
Vulnerability Type	Reflected Cross-Site Scripting (XSS)

Field	Value
Authentication	Not Required

Description

A **Reflected Cross-Site Scripting (XSS)** vulnerability exists in **Online Food Ordering System 1.0** developed by **code-projects.org**.

The vulnerability occurs in the **Order module** within the file:

```
/form/order.php
```

The application fails to properly sanitize the `cust_id` parameter supplied via **HTTP GET requests** before reflecting it in the HTML response.

An attacker can craft a malicious URL containing JavaScript payloads. When a victim visits the crafted link, the injected script executes within the victim's browser context.

Successful exploitation may allow attackers to:

- Steal session cookies
- Hijack authenticated sessions
- Redirect users to malicious websites
- Inject malicious content into the page
- Harvest user credentials

No authentication is required to exploit this vulnerability.

Technical Details

Vulnerable Endpoint

```
/form/order.php
```

Vulnerable Parameter

```
cust_id
```

Request Type

```
GET
```

Vulnerability Class

```
Reflected Cross-Site Scripting (XSS)
```

Root Cause

User-supplied input from the `cust_id` parameter is reflected directly into the HTML response without proper **output encoding or validation**, allowing arbitrary JavaScript execution.

Proof of Concept (PoC)

Malicious URL

The vulnerability can be confirmed by directly accessing the endpoint provided below. No additional reproduction steps are required. The issue has been verified in private browsing mode and across multiple browsers. Ensure that the project path is correctly deployed before the dbfood directory when testing the endpoint.

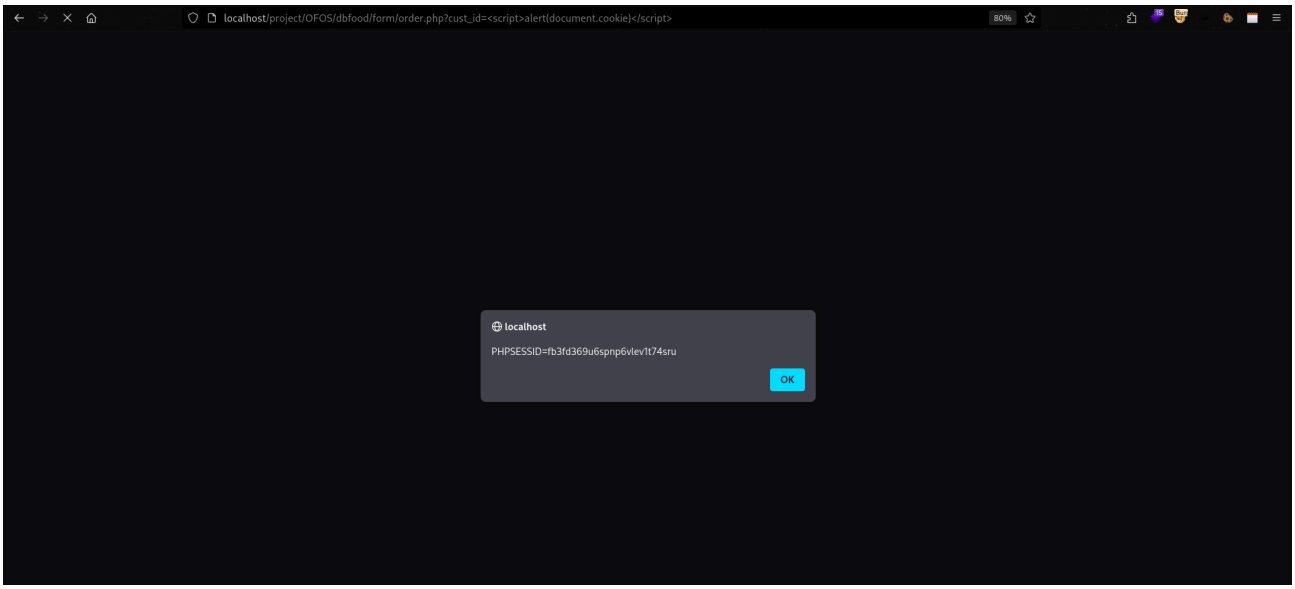
```
http://localhost/dbfood/form/order.php?cust_id=<script>alert(1337)</script>
```

Steps to Reproduce

Step 2. Navigate to the following URL: (Ensure that the project path is correctly deployed before the dbfood directory when testing the endpoint.)

```
http://localhost/dbfood/form/order.php?cust_id=
<script>alert(document.cookie)</script>
```

Ste 3.Upon visiting the crafted URL, the injected JavaScript executes immediately in the browser and displays the document.cookie value, confirming successful XSS execution.



Remediation

1. Output Encoding

Sanitize user input before rendering it in HTML.

```
$cust_id = htmlspecialchars($_GET['cust_id'], ENT_QUOTES, 'UTF-8');
```

---

2. Input Validation

Validate that the parameter contains only expected values.

```php
$cust_id = intval($_GET['cust_id']);
```

---

3. Content Security Policy (CSP)

Implement a restrictive CSP header to mitigate script injection.

```
Content-Security-Policy: default-src 'self'; script-src 'self';
```

---

4. Secure Development Practices

- Never reflect user input directly into HTML responses.
- Use centralized output encoding functions.
- Apply strict input validation for numeric parameters.
