**XSS Cross Site Scripting Vulnerability in \_table\_filter\_dropdown.html at spree Repository**

**Summary:**

Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

|  |
| --- |
| params.to\_unsafe\_h.deep\_merge({q: {shipment\_state\_eq: :shipped, shipment\_state\_not\_in: '', state\_eq: '', state\_in: '', refunded: '', partially\_refunded: ''}}) |

This line directly incorporates user-controlled parameters (params.to\_unsafe\_h) into HTML without sanitization or escaping.

**File Affected:** admin/app/views/spree/admin/orders/\_table\_filter\_dropdown.html.erb  
**Repository:** <https://github.com/spree/spree/blob/main/admin/app/views/spree/admin/orders/_table_filter_dropdown.html.erb>

**Description:**

The provided Ruby on Rails ERB code contains a **Cross-site Scripting (XSS)** vulnerability due to the usage of params.to\_unsafe\_h within the deep\_merge method. This approach allows all incoming parameters—without any filtering or sanitization—to be merged into the query parameters used for rendering order filters in the dropdown menu. Specifically, malicious user input such as <script>alert('XSS')</script> can be injected into parameters like q[shipment\_state\_eq]. Since these parameters are later used in rendering the view without proper encoding or escaping, this opens the door for an attacker to execute arbitrary JavaScript in the context of the user’s browser. This vulnerability arises because to\_unsafe\_h bypasses Rails’ strong parameter protection and trusts the user input blindly, which is a dangerous practice when rendering content in views. If exploited, this could lead to severe consequences such as session hijacking, data theft, or redirection to phishing sites. To fix this, developers should avoid using to\_unsafe\_h and instead whitelist parameters using params.permit(...), ensuring that only expected and safe inputs are processed and displayed. Additionally, all user inputs rendered into the UI must be properly escaped to neutralize any embedded scripts.

**Impact**

XSS (Cross-Site Scripting) can lead to:

* **Data Theft**: Stealing cookies or session tokens.
* **Phishing**: Injecting fake forms or links.
* **Malware**: Distributing harmful software.
* **Defacement**: Altering website appearance.
* **Trust Loss**: Damaging user confidence.

**Proof of Concept (PoC):**

The Proof of Concept is that this line merges all parameters from the incoming request (params.to\_unsafe\_h) directly into a new query hash, which is then passed to link\_to helpers for building URLs in the dropdown menu. The method to\_unsafe\_h converts all parameters, including potentially malicious ones, into a hash without applying any filtering or sanitization.  
?q[shipment\_state\_eq]=<script>alert(1)</script>

**Steps to Reproduce:**

1.   Navigate to the <https://github.com/spree/spree/tree/main>

2.   Go to <https://github.com/spree/spree/blob/main/admin/app/views/spree/admin/orders/_table_filter_dropdown.html.erb>

3.   This line [below] merges all parameters from the incoming request (params.to\_unsafe\_h) directly into a new query hash, which is then passed to link\_to helpers for building URLs in the dropdown menu.

|  |
| --- |
| params.to\_unsafe\_h.deep\_merge({q: {shipment\_state\_eq: :shipped, shipment\_state\_not\_in: '', state\_eq: '', state\_in: '', refunded: '', partially\_refunded: ''}}) |

4.   If an attacker crafts a malicious URL like this:

|  |
| --- |
| https://example.com/admin/orders?q[shipment\_state\_eq]=<script>alert('XSS')</script> |

When a user visits this URL, the controller will receive the parameters with q[:shipment\_state\_eq] equal to <script>alert('XSS')</script>. Because params.to\_unsafe\_h includes this unescaped input and the resulting hash is passed into the link\_to helpers, which generate HTML anchor tags, this script can end up directly embedded in the generated HTML.

**Recommended Fix:**

|  |
| --- |
| Code:  safe\_params = params.permit(q: [:shipment\_state\_eq, :shipment\_state\_not\_in, :state\_eq, :state\_in, :refunded, :partially\_refunded]).to\_h  safe\_query = safe\_params.deep\_merge(...) |

1.   In the **\_table\_filter\_dropdown.html**, this line:

|  |  |
| --- | --- |
| params.to\_unsafe\_h.deep\_merge({q: {shipment\_state\_eq: :shipped, shipment\_state\_not\_in: '', state\_eq: '', state\_in: '', refunded: '', partially\_refunded: ''}}) |  |

* Vulnerable to XSS

2.   In the revised version:

Instead of using params.to\_unsafe\_h, which exposes all raw parameters, the controller should filter only the required and expected values using params.permit(...). For example:

|  |
| --- |
| **safe\_params = params.permit(q: [:shipment\_state\_eq, :shipment\_state\_not\_in, :state\_eq, :state\_in, :refunded, :partially\_refunded]).to\_h**  **safe\_query = safe\_params.deep\_merge(...)** |

3. Finally,

* This ensures that only safe, validated values are included in the URL generation logic, thus preventing any possibility of XSS through URL parameters.