Vulnerability Report

Archivo: dataTables.bootstrap5.min.js

Code Analyzed:

```
/*! DataTables Bootstrap 5 integration
 * 2020 SpryMedia Ltd - datatables.net/license
 */
!function(t){var n,r;"function"==typeof
define&&define.amd?define(["jquery","datatables.net"],function(e){return
t(e,window,document)}):"object"==typeof
exports?(n=require("jquery"),r=function(e,a){a.fn.dataTable||require("datatables.net")(e,a)},"
undefined"!=typeof window?module.exports=function(e,a){return
e=e||window,a=a||n(e),r(e,a),t(a,0,e.document)}:(r(window,n),module.exports=t(n,window,window.
document))):t(jQuery,window,document)}(function(x,e,r,i){"use strict";var
o=x.fn.dataTable;return x.extend(!0,o.defaults,{dom:"<'row'<'col-sm-12 col-md-6'l><'col-sm-12
col-md-6'f>><'row dt-row'<'col-sm-12'tr>><'row'<'col-sm-12 col-md-5'i><'col-sm-12 col-md-
7'p>>",renderer:"bootstrap"}),x.extend(o.ext.classes,{sWrapper:"dataTables_wrapper dt-
bootstrap5",sFilterInput:"form-control form-control-sm",sLengthSelect:"form-select form-
select-sm",sProcessing:"dataTables_processing card",sPageButton:"paginate_button page-
item"}),o.ext.renderer.pageButton.bootstrap=function(d,e,s,a,l,c){function u(e,a){for(var
t,n,r=function(e){e.preventDefault(),x(e.currentTarget).hasClass("disabled")||b.page()==e.data
.action||b.page(e.data.action).draw("page")},i=0,o=a.length;i<o;i++)if(t=a[i],Array.isArray(t)
)u(e,t);else{switch(f=p="",t){case"ellipsis":p="&#x2026;",f="disabled";break;case"first":p=g.s
First,f=t+(0<l?"":" disabled");break;case"previous":p=g.sPrevious,f=t+(0<l?"":"
disabled");break;case"next":p=g.sNext,f=t+(l<c-1?"":"
disabled");break;case"last":p=g.sLast,f=t+(l<c-1?"":"
disabled");break;default:p=t+1,f=l===t?"active":""}p&&(n=-
1!==f.indexOf("disabled"),n=x("<li>",{class:m.sPageButton+" "+f,id:0===s&&"string"==typeof
t?d.sTableId+"_"+t:null}).append(x("<a>",{href:n?null:"#","aria-controls":d.sTableId,"aria-
disabled":n?"true":null,"aria-label":w[t],"aria-role":"link","aria-
current":"active"===f?"page":null,"data-dt-idx":t,tabindex:d.iTabIndex,class:"page-
link"}).html(p)).appendTo(e),d.oApi._fnBindAction(n,{action:t},r))}}var p,f,t,b=new
o.Api(d),m=d.oClasses,g=d.oLanguage.oPaginate,w=d.oLanguage.oAria.paginate||{},e=x(e);try{t=e.
find(r.activeElement).data("dt-idx")}catch(e){}var
n=e.children("ul.pagination");n.length?n.empty():n=e.html("<ul/>").children("ul").addClass("pa
gination"),u(n,a),t!==i&&e.find("[data-dt-idx="+t+"]").trigger("focus")},o});
```

Analysis: ```html
Potential Security Vulnerabilities

Cross-Site Scripting (XSS)

Line: Potentially multiple lines where `d.oLanguage.oPaginate` and `d.oLanguage.oAria.paginate` are used.

Description: The code utilizes user-provided language strings (`d.oLanguage.oPaginate`, `d.oLanguage.oAria.paginate`) to populate the pagination controls. If these strings are not properly sanitized, a malicious user could inject JavaScript code that will be executed in the context of the user's browser. Specifically, the `html(p)` call on line ~62-63 in the pagination renderer is vulnerable.

Mitigation: Sanitize the language strings before using them to generate HTML. Use a proper HTML encoding function to escape any potentially malicious characters. For example, replace `.html(p)` with `.text(p)` if HTML is not required, or use a robust HTML sanitization library if it is. Carefully validate that the strings conform to expected formats. Consider implementing a Content Security Policy (CSP) to further mitigate the impact of XSS attacks.

Code Quality Metrics

Complexity

Description: The `o.ext.renderer.pageButton.bootstrap` function is quite complex, containing nested loops and conditional statements. This increases the cognitive load and makes the code harder to understand and maintain.

Improvement: Break down the function into smaller, more manageable sub-functions. Each sub-function should have a specific responsibility. This improves readability and testability. Consider using more descriptive variable names.

Readability

Description: The use of single-character variable names (e.g., `t`, `n`, `r`, `i`, `o`, `p`, `f`, `b`, `m`, `g`, `w`) makes the code harder to read and understand.

Improvement: Use meaningful variable names that clearly indicate their purpose (e.g., `table` instead of `t`, `paginationElement` instead of `e`). This significantly improves readability and reduces the time required to understand the code.

Coupling

Description: The code is tightly coupled to the DataTables API and Bootstrap's HTML structure. Any changes in either dependency might require significant modifications to this code.

Improvement: Reduce the dependencies on specific DOM element IDs or classes. Abstract the interactions with the DataTables API into a separate module or class to reduce direct dependencies within the renderer.

Proposed Solution

1. **XSS Mitigation:** Implement HTML encoding/sanitization for all user-provided strings used in the pagination rendering logic, specifically around the `html(p)` call. Use `.text(p)` if HTML rendering isn't needed. If HTML rendering is required, utilize a robust HTML sanitization library.
2. **Code Refactoring:** Break down the large `o.ext.renderer.pageButton.bootstrap` function into smaller, more manageable functions with clearly defined responsibilities.
3. **Variable Naming:** Replace single-character variable names with meaningful and descriptive names.
4. **Decoupling:** Reduce direct dependencies on DataTables API and Bootstrap CSS classes by abstracting interactions into separate modules or classes. Consider using template literals to build HTML to better separate code and structure.

```
```