Reporte de Vulnerabilidad

Archivo: dataTables.bootstrap5.min.js

Código Analizado:

```
/*! 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});
```

Análisis: ```html
Identificación de Vulnerabilidades

Posible XSS en la manipulación de DOM

Línea aproximada: 68: `n=e.html("

").children("ul").addClass("pagination")`
Descripción: La función `html()` de jQuery, cuando se usa con contenido proporcionado externamente (aunque aquí parece ser un string constante, es una potencial area a explotar si se modificase), puede ser susceptible a Cross-Site Scripting (XSS) si el contenido HTML inyectado contiene código JavaScript malicioso. Si el valor de `

` llegase a ser dinámico, un atacante podría inyectar scripts maliciosos. Aunque en el estado actual no hay riesgo, es importante prevenir este tipo de vulnerabilidades en el futuro.
Mitigación: Aunque en este caso el contenido parece ser estático, se recomienda utilizar métodos más seguros de manipulación del DOM, como la creación de elementos utilizando `document.createElement()` y la configuración de sus propiedades con `setAttribute()` o `.textContent`. También se podría considerar sanitizar la entrada si esta fuese dinámicamente generada.

Mejora de Métricas: No aplica directamente a métricas de calidad del código en este caso, pero la adopción de prácticas más seguras de manipulación del DOM mejora la mantenibilidad y la seguridad.

getAttribute potencial a XSS

Línea aproximada: 57: `n=x("

- ",{class:m.sPageButton+" "+f,id:0===s&&"string"==typeof t?d.sTableId+"_"+t:null}).append(x("
  ",{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))`

  Descripción: En la construcción de elementos HTML, existe la posibilidad de XSS si las variables `d.sTableId`, `w[t]`, o `p` (dentro de `.html(p)`) contienen código malicioso. Esto es especialmente cierto si estas variables provienen de fuentes no confiables (por ejemplo, entrada del usuario). La concatenación de strings para formar atributos HTML puede llevar a la inyección de HTML/JavaScript.

  Mitigación: * Sanitizar las variables `d.sTableId`, `w[t]`, y `p` antes de usarlas en la construcción de elementos HTML. Utilizar una librería de sanitización o escapar los caracteres especiales HTML. * Usar `.text()` en lugar de `.html()` para insertar contenido de texto, de esta forma se evita la interpretación de HTML en `p`. * Validar y escapar cualquier dato que se use para construir el ID.

  Mejora de Métricas: Reducir el acoplamiento al evitar concatenar strings para formar atributos HTML complejos. Utilizar funciones o métodos que construyan los atributos de manera más segura y legible.

Análisis de Calidad del Código

Complejidad Ciclomática

La función `o.ext.renderer.pageButton.bootstrap` (línea 35) parece tener una complejidad ciclomática relativamente alta debido a la anidación de condicionales (`if`, `switch`) y bucles (`for`). Esto puede dificultar la comprensión y el mantenimiento del código.

Mejora: Descomponer la función en funciones más pequeñas y específicas. Considerar el uso de patrones de diseño como "Strategy" o "Command" para reducir la complejidad de la lógica de paginación.

Legibilidad

El código es relativamente legible, pero algunas partes, como la construcción de elementos HTML en la línea 57, podrían ser más claras. El uso extensivo de abreviaturas (ej. `d`, `e`, `s`, `a`, `l`, `c`, `u`, `p`, `f`, `t`, `b`, `m`, `g`, `w`) dificulta la comprensión rápida del código.

Mejora: Utilizar nombres de variables más descriptivos. Separar la construcción de elementos HTML en pasos más pequeños y comprensibles. Agregar comentarios para explicar la lógica compleja.

Acoplamiento

El código está acoplado a jQuery y DataTables. Esto es inherente a la naturaleza del plugin, pero se podría reducir el acoplamiento al encapsular la lógica específica de jQuery y DataTables en funciones o módulos separados.

Mejora: Definir interfaces claras para la interacción con jQuery y DataTables. Esto facilitaría la sustitución de estas bibliotecas en el futuro.

Duplicación

No se observa una duplicación significativa de código. Sin embargo, la lógica dentro del `switch` (líneas 43-50) podría beneficiarse de una refactorización si se repite en otras partes del código.

Mejora: Extraer la lógica común en funciones reutilizables.

Solución Propuesta

Refactorización con foco en seguridad y legibilidad

El siguiente fragmento ilustra una posible refactorización de la línea 57 (y la lógica circundante) para mejorar la seguridad y la legibilidad. Se enfoca en el uso de métodos seguros de manipulación del DOM y en la claridad del código.

```
function createPageButton(d, t, p, f, l, r, m, g, w, b, e) {
  const isDisabled = f.includes("disabled");
  const li = document.createElement('li');
  li.className = m.sPageButton + " " + f;
  li.id = (0 === s && "string" === typeof t) ? d.sTableId + "_" + t : null;

  const a = document.createElement('a');
  a.href = isDisabled ? null : "#";
  a.setAttribute("aria-controls", d.sTableId);
  a.setAttribute("aria-disabled", isDisabled ? "true" : null);
  a.setAttribute("aria-label", w[t] || ""); // Valor por defecto si no existe
  a.setAttribute("aria-role", "link");
  a.setAttribute("aria-current", "active" === f ? "page" : null);
  a.setAttribute("data-dt-idx", t);
  a.tabIndex = d.iTabIndex;
  a.className = "page-link";
  a.textContent = p; // Usar textContent para evitar XSS

  li.appendChild(a);
  e.appendChild(li); // 'e' representa el elemento al que se adjunta el botón

  d.oApi._fnBindAction(li, { action: t }, r);
}

//Uso dentro de la función bootstrap
//Reemplazando la linea 57 y adaptando los parametros
createPageButton(d, t, p, f, l, r, m, g, w, b, n);
```

Explicación: * Se crea una función `createPageButton` para encapsular la lógica de creación de los botones de paginación. * Se utiliza `document.createElement()` para crear los elementos HTML de forma segura. * Se usa `setAttribute()` para establecer los atributos de los elementos. * Se utiliza `textContent` en lugar de `html()` para insertar el texto del botón, previniendo XSS. * Se pasan los parámetros necesarios a la función para que pueda ser reutilizada.

```
```