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
Análisis de Vulnerabilidades y Calidad del Código
Posible Vulnerabilidad XSS

Descripción

El código manipula HTML directamente usando `x("").html(p)`. Si la variable `p` contiene HTML malicioso (por ejemplo, desde `g.sFirst`, `g.sPrevious`, `g.sNext`, `g.sLast`, o incluso directamente desde el número de página `t+1`), podría ser explotado para Cross-Site Scripting (XSS). Esto se vuelve especialmente crítico si las variables de idioma (`g.sFirst`, etc.) son configurables por el usuario o provienen de una fuente no confiable. El manejo de `aria-label` también usa `w[t]`, que podría ser una fuente de XSS si `w` se construye de manera insegura.

Línea Aproximada: Dentro de la función `u`, cerca de la creación del elemento ``:

```
x("",{/*...*/}).html(p)
```

Mitigación

1. **Escapado de HTML:** Siempre escapar el contenido de la variable `p` antes de insertarlo en el HTML. Usar, por ejemplo, `x("").text(p)` en lugar de `.html(p)` cuando `p` solo deba contener texto. Si `p` *debe* contener HTML, sanitizarlo usando una librería de sanitización HTML confiable.
2. **Sanitización de Datos de Idioma:** Asegurarse de que los valores en `g.sFirst`, `g.sPrevious`, `g.sNext`, `g.sLast`, y `w` estén correctamente escapados o sanitizados al momento de ser definidos y almacenados, antes de que sean usados por este código.

3. **Validación de Entrada:** Si los números de página son derivados de entrada del usuario, validar y sanitizar rigurosamente esa entrada para prevenir la inyección de código.

Métricas de Calidad del Código

Complejidad Ciclomática

La función `o.ext.renderer.pageButton.bootstrap` parece tener una complejidad ciclomática alta debido a la presencia de múltiples condicionales (`if`, `switch`) y bucles (`for`). Esto dificulta la comprensión, prueba y mantenimiento del código.

Acoplamiento

El código está fuertemente acoplado a jQuery y a la API interna de DataTables (`d.oApi._fnBindAction`). Esto dificulta su reutilización en otros contextos y hace que el código sea más sensible a los cambios en DataTables.

Legibilidad

La legibilidad se ve comprometida por el uso de nombres de variables abreviados (por ejemplo, `d`, `e`, `s`, `a`, `l`, `c`, `u`, `p`, `f`, `t`, `b`, `m`, `g`, `w`). Una mejor elección de nombres mejoraría la comprensión del código.

Duplicación

Hay una pequeña duplicación en el manejo de las clases "disabled" en los casos "first", "previous", "next", y "last".

Mejoras Sugeridas

- **Refactorización:** Dividir la función `o.ext.renderer.pageButton.bootstrap` en funciones más pequeñas y especializadas para reducir la complejidad.
- **Nombres Descriptivos:** Usar nombres de variables más descriptivos.
- **Abstracción:** Considerar la posibilidad de abstraer la lógica de renderizado de los botones de paginación para reducir el acoplamiento.
- **Eliminación de Duplicación:** Refactorizar el código para eliminar la duplicación en el manejo de la clase "disabled". Por ejemplo, crear una función auxiliar que determine si un botón debe estar deshabilitado o no.

Solución Propuesta (Fragmento)

El siguiente fragmento ilustra cómo se podría escapar el contenido de la variable `p` para mitigar el riesgo de XSS. Además, se renombran variables para mayor claridad.

```
function u(element, buttons) {
  for (let i = 0; i  buttons.length; i++) {
    const button = buttons[i];
    if (Array.isArray(button)) {
      u(element, button);
    } else {
      let buttonContent = '';
      let buttonClass = '';

      switch (button) {
        case "ellipsis":
          buttonContent = "…";
          buttonClass = "disabled";
          break;
        case "first":
          buttonContent = language.oPaginate.sFirst;
          buttonClass = button + (l > 0 ? "" : " disabled");
          break;
        // ... other cases ...
        default:
          buttonContent = button + 1;
          buttonClass = l === button ? "active" : "";
```

```
      }

      if (buttonContent) {
        const isDisabled = buttonClass.indexOf("disabled") !== -1;
        const buttonElement = x("
• ", {
          class: classes.sPageButton + " " + buttonClass,
          id: 0 === s && "string" == typeof button ? tableSettings.sTableId + "_" + button :
null
        }).append(
            x("", {
              href: isDisabled ? null : "#",
              "aria-controls": tableSettings.sTableId,
              "aria-disabled": isDisabled ? "true" : null,
              "aria-label": ariaLabels[button],
              "aria-role": "link",
              "aria-current": "active" === buttonClass ? "page" : null,
              "data-dt-idx": button,
              tabindex: tableSettings.iTabIndex,
              class: "page-link"
            }).text(buttonContent) // Use .text() instead of .html()
        ).appendTo(element);

        tableSettings.oApi._fnBindAction(buttonElement, { action: button }, function (e) {
          e.preventDefault();
          if (!x(e.currentTarget).hasClass("disabled")) {
            api.page(e.data.action).draw("page");
          }
        });
      }
    }
  }
}
```

Cambios Clave: Se utiliza `.text(buttonContent)` en lugar de `.html(p)` para insertar el contenido del botón, lo que previene la inyección de HTML malicioso. Además, las variables se han renombrado para mejorar la legibilidad.

```
```