Relative Path Traversal Attack on note creation

Moderate **davidmehren** published **GHSA-p528-555r-pf87** on Apr 26, 2021

Package

hedgedoc

Affected versions	Patched versions
< 1.8.0	1.8.0

Description

Impact

An attacker can read arbitrary `.md` files from the server's filesystem due to an improper input validation, which results in the ability to perform a relative path traversal.

CVSSv3 string: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

PoC / Quicktest

To verify if you are affected, you can try to open the following URL: `http://localhost:3000/..%2F..%2FREADME#` (replace `http://localhost:3000` with your instance's base-URL e.g. `https://demo.hedgedoc.org/..%2F..%2FREADME#`).

- If you see a README page being rendered, you run an affected version.

Analysis

The attack works due the fact that the internal router passes the url-encoded alias to the `noteController.showNote` -function. This function passes the input directly to `findNote()` utility function, that will pass it on the the `parseNoteId()` -function, that tries to make sense out of the noteId/alias and check if a note already exists and if so, if a corresponding file on disk was updated.

If no note exists the note creation-function is called, which pass this unvalidated alias, with a `.md` appended, into a `path.join()` -function which is read from the filesystem in the follow up routine and provides the pre-filled content of the new note.

This allows an attacker to not only read arbitrary `.md` files from the filesystem, but also observes changes to them.

The usefulness of this attack can be considered limited, since mainly markdown files are use the file-ending `.md` and all markdown files contained in the hedgedoc project, like the README, are public anyway. If other protections such as a chroot or container or proper file permissions are in place, this attack's usefulness is rather limited.

Patches

```
diff --git a/lib/models/note.js b/lib/models/note.js
index 9fe02359..49b7ce84 100644
--- a/lib/models/note.js
+++ b/lib/models/note.js
@@ -96,7 +96,7 @@ module.exports = function (sequelize, DataTypes) {
             if (!note.alias) {
               filePath = config.defaultNotePath
             } else {
-              filePath = path.join(config.docsPath, note.alias + '.md')
+              filePath = path.join(config.docsPath, path.basename(note.alias) + '.md')
             }
             if (Note.checkFileExist(filePath)) {
               var fsCreatedTime = moment(fs.statSync(filePath).ctime)
@@ -195,7 +195,7 @@ module.exports = function (sequelize, DataTypes) {
           }
         }).then(function (note) {
           if (note) {
-            let filePath = path.join(config.docsPath, noteId + '.md')
+            let filePath = path.join(config.docsPath, path.basename(noteId) + '.md')
             if (Note.checkFileExist(filePath)) {
               // if doc in filesystem have newer modified time than last change time
               // then will update the doc in db
@@ -237,7 +237,7 @@ module.exports = function (sequelize, DataTypes) {
               return callback(null, note.id)
             }
           } else {
-            var filePath = path.join(config.docsPath, noteId + '.md')
+            var filePath = path.join(config.docsPath, path.basename(noteId) + '.md')
             if (Note.checkFileExist(filePath)) {
               Note.create({
                 alias: noteId,
```

Workarounds

On a reverse-proxy level one can force a URL-decode, which will prevent this attack because the router will not accept such a path.

For more information

If you have any questions or comments about this advisory:

- Open an topic on our community forum
- Join our matrix room

Severity

Moderate

CVE ID

CVE-2021-29474

Weaknesses

CWE-20 CWE-23