Symlink Resolution #225

Answered by trs

nfacha asked this question in Q&A

nfacha on Dec 15, 2020

You can escape root if there is a symlink withing the FTP, so creating a symlink inside the root such as `ln -s / x` will allow you to access the server root

↑ 2

Answered by **trs** on Dec 17, 2020

This library enables creating a custom filesystem, you can use this to enable checking symlinks.

You could overwrite the `_resolvePath` and perform a check there. Here's a quick example, not tested and probably doesn't work, but it gives you the idea.

```
// symlinkCheckingFileSystem.js
```

View full answer ↓

11 suggested answers

Oldest | Newest | Top

trs on Dec 15, 2020 edited

Node's `path` module doesn't resolve symlinks, so it won't change the directory that this package is using.
Should we disallow `cd` -ing into a symlinked directory? ~~If so adding a `stat.isSymbolicLink()` check to the `chdir` command would be the place to check.~~ This doesn't take into account cd-ing into `link/dir` as `dir` is not a symlink.

↑ 1

0 replies

nfacha on Dec 15, 2020 Author

> Node's `path` module doesn't resolve symlinks, so it won't change the directory that this package is using.
> Should we disallow `cd` -ing into a symlinked directory? If so adding a `stat.isSymbolicLink()` check to the `chdir` command would be the place to check.

Im able to navigate thru symlinks using this (and escape root doing so), it should be diisabled by default IMHO

↑ 1

0 replies

nfacha on Dec 15, 2020 Author

Tried making these changes, although im still able to go thru the symlink and escape root with these changes (server on Linux, connecting thru Filezilla), am I missing something obvious?

5 src/fs.js

```
@@ -57,6 +57,9 @@ class FileSystem {
          return fsAsync.access(filePath, constants.F_OK)
          .then(() => {
            return fsAsync.stat(filePath)
+            .tap((stat) => {
+              if (stat.isSymbolicLink()) throw new errors.FileSystemError('Not a valid directory');
+            })
            .then((stat) => _.set(stat, 'name', fileName));
          })
          .catch(() => null);
@@ -69,7 +72,7 @@ class FileSystem {
      const {fsPath, clientPath} = this._resolvePath(path);
      return fsAsync.stat(fsPath)
      .tap((stat) => {
-       if (!stat.isDirectory()) throw new errors.FileSystemError('Not a valid directory');
+       if (!stat.isDirectory() || stat.isSymbolicLink()) throw new errors.FileSystemError('Not a valid directory');
      })
      .then(() => {
        this.cwd = clientPath;
```

Added the check on both the chdir and list commands

↑ 1

0 replies

matt-forster on Dec 15, 2020 edited

Unless the client can create that symlink, the fact that the target folder is a link is a non-issue and should be the same as #167. You wouldn't be able to escape out of the linked folder into *that* location, it would still resolve to the location of the local filesystem.

For example a file system like this;

```
/
- tmp/
-- ftp
--- user1/
---- shared/ _linked to /tmp/link/shared/_
--- user2/
---- shared/ _linked to /tmp/link/shared/_
-- link/
--- shared/
---- sites/
```

and a server setup as:

```
root = '/tmp/ftp/user2'
cwd = '/'
```

The paths should resolve as;
`/ -> /tmp/ftp/user2`
`/shared -> /tmp/ftp/user2/shared` linked to `/tmp/link/shared`
`/shared/sites/ -> /tmp/ftp/user2/shared/sites` linked to `/tmp/link/shared/sites`
`/shared/sites/../../ -> /tmp/ftp/user2` (not `/tmp/link/`)

The fact that the FS is setup to *use* symlinks isn't the issue of the library, and I think is legitimate.

@nfacha Am I describing what is happening here correctly? You are able to say, navigate to `/tmp/link` through the symlink?

↑ 1 👍 1

0 replies

matt-forster on Dec 15, 2020

@nfacha Can you checkout #224 and see if this resolves the issue you are having?

↑ 1

0 replies

trs on Dec 15, 2020

Symlinks can still be followed through, but can't escape in the way @forstermatth described above.

↑ 1

0 replies

nfacha on Dec 16, 2020 Author

edited

> **@nfacha** Can you checkout #224 and see if this resolves the issue you are having?

After checking out #224 i can still escape root on the following setup using Fllezilla
User root: /home/fmcs
Symlink at /root/fmcs named x created with `ln -s / x`

If I switch into the symlink i can still see the server root and navigate thru the folders (that are outside the user root)

From the PR changes it looks like you are appending the root path on the beginning, this is not enough tho, as going thru a symlink resolves the path with the root on the beginning already, after adding some logging(print out `fsPath` from `_resolvePath`) I can see that if I go thru the symlink to `/x/etc/apt` the path gets resolved to `/home/fmcs/x/etc/apt` and that is technically under the path so it allows me to go in there and I can still escape root, the only solution I can see is to check if any of the directories used on the path is a symlink?

↑ 1

0 replies

nfacha on Dec 16, 2020 Author

I have sent in a PR (#226) that fixes root escaping thru symlinks.

I don't usually work with node, so you might want to do some adjustments as while it works likely there is a better way to do it.

↑ 1

0 replies

matt-forster on Dec 16, 2020

edited

So, at the moment, we are not in agreement that this is an issue this library needs to be responsible for.

My opinion is that, to prevent this issue, do not create the symlink in your filesystem.
The library doesn't really take responsibility for the FS that you present to your users.

You are actively creating that symlink and making it available to your user. As an admin, there are legitimate uses for symlinks, and I think it is the server's responsibility to make sure the filesystem you are using is set up correctly. FTP clients cannot create these symlinks, so I don't think there is a security issue here.

↑ 1

0 replies

trs on Dec 17, 2020

This library enables creating a custom filesystem, you can use this to enable checking symlinks.

You could overwrite the `_resolvePath` and perform a check there. Here's a quick example, not tested and probably doesn't work, but it gives you the idea.

```
// symlinkCheckingFileSystem.js
const {FileSystem} = require('ftp-srv');
const isPathInside = require('is-path-inside'); // https://github.com/sindresorhus/is-path-inside
const {realPathSync} = require('fs');

module.exports.SymlinkCheckingFileSystem = class SymlinkCheckingFileSystem extends FileSystem {
  constructor() {
    super(...arguments);
  }

  _resolvePath(path) {
    // Let base resolver do it's thing
    const {clientPath, fsPath} = super._resolvePath(path);

    // Resolve symlinks in path
    const realFsPath = realPathSync(fsPath);

    if (isPathInside(realFsPath, this.root)) {
       // Oh no, this path escapes root!
       throw new Error('Tsk tsk');
    }

    return {clientPath, fsPath};
  }
}



// index.js
const {FtpSrv} = require('ftp-srv');
const {SymlinkCheckingFileSystem} = require('./symlinkCheckingFileSystem.js');

const ftpServer = new FtpSrv();

ftpServer.on('login', (connection, resolve, reject) => {
  // ...
  resolve({
    fs: new SymlinkCheckingFileSystem(connection, {root: '...', cwd: '...'}),
    // ...
  });
});

// ...

ftpServer.listen()
```

This isn't something we want to do in the base library as `_resolvePath` must work with virtual file systems. Doing this check restricts us to the actual file system. So a custom implementation would suite best.

Marked as answer · 1 · 1

0 replies

Answer selected by **matt-forster**

matt-forster on Dec 19, 2020 · edited

@Heartz66 Please keep the discussion about symlinks here.

> Please take a look at the code of the pterodactyl daemon: pterodactyl/daemon@ `208f85c` /src/controllers/server.js#L579

> They use fs.realpathSync to resolve symlinks and get the exact path of the path input. As last measure they check if the result path starts with the root value of the folder. The current measures by ftp-srv do not protect against symlinks.

I'd like some more information as to the relevance of the pterodactyl game server, in regards to resolving symlinks in the context of an extensible FTP server. We have avoided making assumptions around the structure of the filesystem the user wants to expose, which can legitimately include symlinks (see above for an example). It is possible to write a resolution extension (example above) that removes the use of symlinks outside of the defined user root. But as far as we can tell, there is no reason to completely block the use of symlinks.

1

0 replies

Category

Q&A

Labels

None yet

3 participants

Converted from issue

This discussion was converted from issue #225 on December 28, 2020 20:23.