How to prevent path escapes in rust?

help

SpiderUnderUrBed July 31, 2025, 11:15pm 1

In axum i am making it so a request is sent to the backend for a list of files at a given directory, how do i prevent escapes? This is for a filebrowser website im making (self hosted). So I want to confine the filebrowser view to a certain directory, this means the frontend sending requests to the backend for a specific path, however with .. and other forms of escapes, I want to prevent escapes.

i simply want to list directories. So im not serving html and stuff there, im allowing a way to interact with a folder over api calls, like list the directories, send api calls to modify files and folders. Not looking for servedir

simonbuchan July 31, 2025, 11:58pm 2

What are you using? read_dir doesn't return the current or parent dir.

It's not clear what you mean by “path escapes” either - symbolic links? Hidden files? Some form of directory traversal concern?

SpiderUnderUrBed August 1, 2025, 12:04am 3

Symlinks, and .. or . escapes, but I feel like path escapes extend far beyond those too. Directory traversal is my concern

simonbuchan August 1, 2025, 12:18am 4

Directory traversal is generally handled by resolving the absolute path then checking the root is a prefix, eg with Path::starts_with

Still not sure what you mean by “escape” here if not a traversal attack. That's not a directory entry type for example.

quinedot August 1, 2025, 1:02am 5

There's a couple different concerns, IMO.

One is input sanitation: getting a request that tries to escape some area using `..` or similar. For that, normalize_lexically may someday be useful if you want to be lenient. Today, you could look at the components and reject anything besides `Normal` and perhaps `RootDir`, if you don't care about being lenient.

Another is if you can't trust the filesystem contents: symlinks and perhaps bind mounts that lead outside an expected directory hierarchy. There are various things you could do here, like walking the path yourself or using `canonicalize` and making sure it starts under the expected path. They're all TOCTOU to various degrees since the filesystem may change before you actually open a directory. They may still be useful for accidental or latent forms of escapes, but they aren't bulletproof.

1 Like

simonbuchan August 1, 2025, 4:05am 6

You could maybe handle that by opening then doing a `readlink("/proc/self/fd/{fd}")` then comparing dev/inodes, but I haven't run through the logic properly.

Maybe Windows was right to not have symlinks

quinedot August 1, 2025, 6:49am 7

It's not just symlinks, it's also things like an open directory getting moved. There are `libc` ways around both of those through the point of getting a file descriptor (no need to rely on proc). And perhaps crates, but not enough in `std`.

(I'm speaking from a linux POV. I don't know the considerations for Windows.)

simonbuchan August 1, 2025, 8:23am 8

Reading /proc/self/fd gives you the path that an fd was opened with, so you could open then resolve then check the prefix. But it only gives you the path at the time it was opened, so I didn't know that it actually helps much..

The main differences with Windows FS as I understand it is the default rule there is once you open a file it can't be moved, partially because the file identity is the file path, rather than an inode. It's generally a lot easier to avoid TOCTOU, in my experience.

bjorn3 August 1, 2025, 8:39am 9

Dir in cap_std::fs - Rust may be useful for you.

system Closed October 30, 2025, 8:39am 10

This topic was automatically closed 90 days after the last reply. We invite you to open a new topic if you have further questions or comments.

Related topics

Topic		Replies	Views	Activity
Anything in hyper_staticfile to avoid "../../../" attacks?	3	461	May 2, 2022	
Recursive directory traversal announcements	1	3430	January 12, 2023	
Add "Cli style" relative path to a pathbuf help	10	203	October 14, 2025	
Fs::walk_dir causes too many open files help	3	1425	January 12, 2023	
Thinking of reimplementing a filesystem abstraction API help	2	414	April 29, 2023	