

[New issue](#)[Jump to bottom](#)

# Sanic static handler allows parent ".." directory traversal #2478

🔒 Closedekzhang opened this issue on Jun 10 · 4 comments · Fixed by [#2495](#)

ekzhang commented on Jun 10 • edited ▼

## Describe the bug

The `sanic static` directory code checks for `../` as a substring of paths, but it also unquotes the path, which allows a malicious user to escape outside the static folder by using `..%2F`, where `%2F` is the URL-escaped version of `/`.

## Code snippet

First, a basic server called `main.py`.

```
from sanic import Sanic

app = Sanic(name="sanic_test")

app.static('/static', './static_files')

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

Then create a static file folder.

```
mkdir static_files
cat "hello world" > static_files/a.txt
```

Now run the server with `python3 main.py` and:

```
$ curl http://localhost:8000/static/a.txt
hello world
$ curl http://localhost:8000/static/..%2Fstatic_files/a.txt
hello world
```

This is very surprising behavior. From a security perspective it is not critical because sanic checks that the final resolved path has a prefix with the static directory, but this allows an attacker to expose information like the name of the static file folder.

Another case where this is dangerous is if you have a middleware that only allows a user to see certain subpaths like `/static/public/**` of the `/static/**` routes without authentication. Then, even without authentication, a user could visit a path like `/static/public/..%2Fprivate/secret_content.txt` and retrieve the contents of `/static/private/secret_content.txt`.

### Expected behavior

Sanic should not allow parent directory traversal in static folders.

### Environment (please complete the following information):

- OS: macOS and Linux
- Version 22.3.2

Tronic commented on Jun 10

Member

Related to [#2477](#).

I suppose this can also be exploited by using backslashes, which on Windows would be interpreted as path separators. I suggest denying any escaped slashes and plain backslashes. Processing of `..` (and `.`) ought to be preserved in a logical form, possibly yielding a redirect response, but never traversing parents on the actual filesystem. Somewhat related, all path elements that begin with a dot could be banned (avoiding access to hidden files, which also could be leaking data not intended public).

Python `pathlib` module would be a reasonable option for path processing, avoiding Sanic's own parsing, but concerns include performance and the said special cases with slashes being escaped. Also, whether URL sanitation (avoiding `.`, `..`, `//` and other oddities) should be applied on all requests and not only static files.

Just my 5 cents. I didn't look at the current code, and in any case this probably needs more thought for a solid implementation.

ekzhang commented on Jun 11

Author

Most browsers and HTTP clients (like cURL) will automatically apply [path normalization](#) to URLs, which includes removing dot segments. It's not really necessary to process `..` as a result.



1

prryplatypus commented on Jul 10 • edited ▼

Member

Funlily enough it looks like we already do some kind of check, but we do so before unquoting it :S  
EDIT: I misunderstood the issue, my bad!

  prryplatypus mentioned this issue on Jul 10


## Prevent directory traversal with static files #2495

 Merged

prryplatypus commented on Jul 10 • edited ▼

Member

Ok, I've given it a try and have ended up with this. Quick things to note:

- I have *not* benchmarked it at all, so if anyone is able to do that I'd be more than grateful!
- I believe it should fix both quoted and unquoted traversals (you can try this with cURL by passing `--path-as-is` ).
- I'm unsure whether lines 827-836 are needed anymore. Feedback welcome.
- It *might* fix  [Sanic static directory fails when folder name ends with ".." #2477](#) too? I haven't tried it, but wondering if it may have done so as a side effect.

 ahopkins closed this as completed in [#2495](#) on Jul 28

  GoVulnBot mentioned this issue on Aug 1

x/vulndb: potential Go vuln in [github.com/sanic-org/sanic](https://github.com/sanic-org/sanic): CVE-2022-35920  
golang/vulndb#757

 Closed

  jba mentioned this issue on Aug 1

x/vulndb: potential Go vuln in [github.com/sanic-org/sanic](https://github.com/sanic-org/sanic): CVE-2022-35920 jba/nested-modules#385

 Open

### Assignees

No one assigned

### Labels

None yet

---

Projects

None yet

---


Milestone

No milestone

---

Development

Successfully merging a pull request may close this issue.

 **Prevent directory traversal with static files**  
sanic-org/sanic

---

3 participants

