# Notepad

I examined the tarball that was provided.  The static folder is empty. The template folder contains template HTML webpages.  The index file is interesting. It generates a webpage for us based on the error we provide it with. It also allows us to view the error file with the same name stored in the error directory that we can't usually access.

On checking out the docker file, I can see the flag.txt being mentioned. So we need to access it from within the application. It seems the file is being renamed to the uuid 😕

In the python program, it prevents us from entering characters like _ or /.

It renames the note to the first few characters of the content and appends a random token. The url_fix function also plays a role here. We can use that to bypass the badcontent error.

```
@app.route("/")

def index():

    return render_template("index.html", error=request.args.get("error"))
```

With this we can pass arguments through the url. This will help us access our error.

We can inject our own malicious code in the contents of the note and save it into the error folder. This can later be accessed by URL arguments.

I'll input this text. I used a random string generator to generate the rest of the text.

..\templates\errors\mgvzfteoumcojqjtfmtpqlhwqigdmppuwwjwwvodjvbmkobxbahuvcdqdhhpiavyxyldsewhdvegijgvffmennhgcbyhhilswulscpnljzea

This generates the error file where we can send our payload and view code execution. We can view it by using the ?erorr=  flag in the url and adding the error file name to the URL.

We can send the following code to gain access:

*{{ ''[request.args.get('class')].mro()[1][request.args.get('subclasses')]()[265](['cat', 'flag-c8f5526c-4122-4578-96de-d7dd27193798.txt'], stdout=-1).communicate() }}*

*This will print the value of the flag file*

## Reference

https://werkzeug.palletsprojects.com/en/2.2.x/urls/#werkzeug.urls.url_fix

https://medium.com/@nyomanpradipta120/ssti-in-flask-jinja2-20b068fdaeee