LEVEL3 - Cybersecurity
pyjail - portal

Matthias Naumann - 42 Heilbronn
github.com/MaNafromSaar

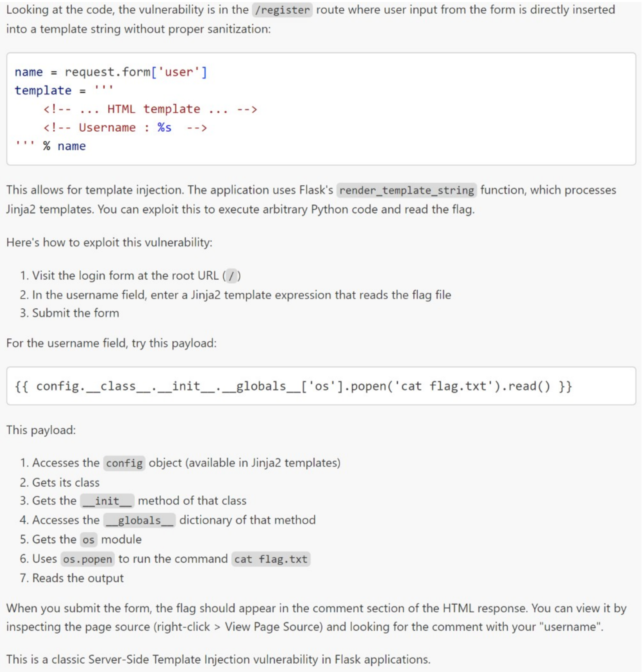
gist:
Break a webshell via SSTI to get the flag.

Steps:
1. run the python app
2. open the corresponding website via browser
3. inspect
4. try ssti
5. get the flag

Ok, this one was easy with some help. Running the app and opening the corresponding site is straightforward.
A first inspection of the site showed a field "for debugging use only".
Already the first attempt:

```
      {{ config.__class__.__init__.__globals__['os'].popen('cat flag.txt').read() }}
```

yielded the flag being displayed in the said debugging part. I found this command asking GitHub Copilot Pro with Claude 3.7 Sonnet in agent mode.



Looking at the code, the vulnerability is in the `/register` route where user input from the form is directly inserted into a template string without proper sanitization:

```
name = request.form['user']
template = '''
    <!-- ... HTML template ... -->
    <!-- Username : %s  -->
''' % name
```

This allows for template injection. The application uses Flask's `render_template_string` function, which processes Jinja2 templates. You can exploit this to execute arbitrary Python code and read the flag.

Here's how to exploit this vulnerability:

1. Visit the login form at the root URL (`/`)
2. In the username field, enter a Jinja2 template expression that reads the flag file
3. Submit the form

For the username field, try this payload:

```
{{ config.__class__.__init__.__globals__['os'].popen('cat flag.txt').read() }}
```

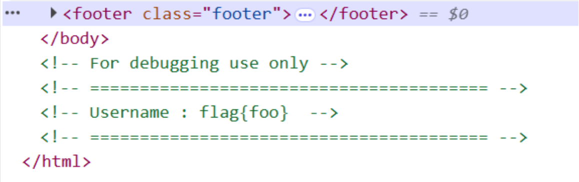
This payload:

1. Accesses the `config` object (available in Jinja2 templates)
2. Gets its class
3. Gets the `__init__` method of that class
4. Accesses the `__globals__` dictionary of that method
5. Gets the `os` module
6. Uses `os.popen` to run the command `cat flag.txt`
7. Reads the output

When you submit the form, the flag should appear in the comment section of the HTML response. You can view it by inspecting the page source (right-click > View Page Source) and looking for the comment with your "username".

This is a classic Server-Side Template Injection vulnerability in Flask applications.

The way via SSTI was clear to me from the start but getting the specific command to attack this constellation was provided by AI, I must admit. So as predicted, the flag was displayed in the debugging comment section:



```
•••   ▸<footer class="footer">⋯</footer> == $0
    </body>
    <!-- For debugging use only -->
    <!-- ======================================== -->
    <!-- Username : flag{foo}  -->
    <!-- ======================================== -->
  </html>
```

flag{foo}

PS: If it wasn't via asking AI, I would have checked repos and sites with lists of SSTI Payloads - e.g. https://www.onsecurity.io/blog/server-side-template-injection-with-jinja2/