Introduction to Template Engines

Template engines read tokenized strings from template documents and produce rendered strings with actual values in the output document. Templates are commonly used as an intermediary format by web developers to create dynamic website content. Server-Side Template Injection (`SSTI`) is essentially injecting malicious template directives inside a template, leveraging Template Engines that insecurely mix user input with a given template.

Below you will find some applications that you can run locally to better under templates. If you are unable to do so, do not worry. The following sections feature exercises with various applications utilizing templates.

Let us now consider the following documents:

app.py

Code: python

```python
#/usr/bin/python3
from flask import *

app = Flask(__name__, template_folder="./")

@app.route("/")
def index():
    title = "Index Page"
    content = "Some content"
    return render_template("index.html", title=title, content=content)

if __name__ == "__main__":
    app.run(host="127.0.0.1", port=5000)
```

index.html

Code: html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>{{title}}</h1>
    <p>{{content}}</p>
</body>
</html>
```

When we visit the website, we will receive an HTML page containing the values of the `title` and `content` variables evaluated inside the double brackets on the template page. Pretty straightforward, and as we can see, the user does not have any control over the variables. What happens when user input enters a template without any validation, though?

app.py

Code: python

```python
#/usr/bin/python3
from flask import *

app = Flask(__name__, template_folder="./")

@app.route("/")
def index():
    title = "Index Page"
    content = "Some content"
    return render_template("index.html", title=title, content=content)

@app.route("/hello", methods=['GET'])
def hello():
    name = request.args.get("name")
    if name == None:
        return redirect(f'{url_for("hello")}?name=guest')
    htmldoc = f"""
    <html>
    <body>
    <h1>Hello</h1>
    <a>Nice to see you {name}</a>
    </body>
    </html>
    """
    return render_template_string(htmldoc)

if __name__ == "__main__":
    app.run(host="127.0.0.1", port=5000)
```

In this case, we can inject a template expression directly, and the server will evaluate it. This is a security issue that could lead to remote code execution on the target application, as we will see in the following sections.

cURL - Interacting with the Target

cURL - Interacting with the Target

```
Govardhan Gujji22@htb[/htb]$ curl -gis 'http://127.0.0.1:5000/hello?name={{7*7}}'

HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 79
Server: Werkzeug/2.0.2 Python/3.9.7
Date: Mon, 25 Oct 2021 00:12:40 GMT


    <html>
    <body>
    <h1>Hello</h1>
    <a>Nice to see you 49</a> # <-- Expresion evaluated
    </body>
    </html>
```

Previous Next

Mark Complete & Next

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

Powered by HACKTHEBOX