EVIL-USER-STORY (d'après la proposition de Yannick MIDEY)

Basé sur le modèle recto-verso de
https://www.riskinsight-wavestone.com/2019/12/cybersecurity-transformation-agile/

"***En tant que*** **personne malveillante, *je veux* avoir accès à la base de données *afin d'*extraire, pour exploitation malveillante, le login et le mot de passe des utilisateurs de l'application Vetux-Line"**

Contre-mesure : En tant que développeur, afin d'empêcher les personnes malveillantes qui souhaitent à partir de la base de données accéder aux compte des utilisateurs de l'application Vetux-Line, les mots de passe seront chiffrés. Pour cela j'utilise une classe de l'interface *UserPasswordEncoderInterface.* En effet, de telles classes possèdent la fonction encodePassword qui permet d'encoder le mot de passe afin qu'il ne soit plus en clair. Dans mon projet, l'utilisateur est
manuellement créé dans le fichier AppFixtures. Exemple :

```php
<?php

# Nom du package
namespace App\DataFixtures;

# Importation des classes nécessaires
use App\Entity\Admin;
use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Persistence\ObjectManager;
use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;

# Nom de la classe
class AppFixtures extends Fixture
{
    # Déclaration des attributs
    private $passwordEncoder;

    # On encode l'attribut passwordEncorder avec un système de hachage provenant
    # de la classe UserPasswordEncoderInterface de la librairie doctrine
    public function __construct(UserPasswordEncoderInterface $passwordEncoder){
        $this->passwordEncoder = $passwordEncoder;
    }

    public function load(ObjectManager $manager)
    {
        # On appelle la classe Admin situé dans App\Entity\Admin
        $user = new Admin();
        # Appel de la méthode setUsername avec en paramètre la chaine de
        # caractère root de l'objet référencé par user
        $user->setUsername("root");
        # Appel de la méthode setPassword avec en paramètre l'attribut passwordEncoder et avec co
        $user->setPassword($this->passwordEncoder->encodePassword($user, "sio"));
        # On notifie doctrine avec la méthode persist en lui disant que
        # l'on voudrait ajouter les valeurs que contient la variable $user
        # dans la base de données
        $manager->persist($user);
        # Avec la méthode flush de doctrine, on met à jour notre base de données
        $manager->flush();
    }
}
```

Dans la table `admin` de ma base de données, le mot de passe du compte root n'est plus affiché en claire, donc affichée "sio" mais il est affiché crypté.

La personne malveillante même en réussissant à récupérer le login et le mot de passe entrée dans la base de donnée ne pourra donc pas avoir un accès à l'application.