

Evil Storie Vetux-Line

Participant :
Enzo Carpentier
Corentin Bonifacio
Killian Moliere

Evil Storie 1 :

En tant que personne malveillant, je voudrais avoir accès a la page admin ou une autre page que je ne suis pas sensé avoir accès avec mon simple ROLE d'utilisateurs, je vais donc essayé de bidouillé l'URL pour avoir accès sans la permission ou autre moyen..

En tant que développer voulant contré cela je vais donc a chaque page qui requiert une sécurisation en fonction du ROLE voici donc le code que je dois ajouter :

```
/**
 * @return Collection/Role[]
 */
public function getRole(): Collection
{
    return $this->role;
}

public function addRole(Role $role): self
{
    if (!$this->role->contains($role)) {
        $this->role[] = $role;
    }

    return $this;
}

public function removeRole(Role $role): self
{
    $this->role->removeElement($role);

    return $this;
}
```

Ainsi tous ceux qui ont pas la permission des rôles ne pourront pas accéder au page que nous auront configurez pour en y ajouté une petite sécurisation sur les fichiers twigs :

Pour y avoir accès seulement si on est GESTIONNAIRE :

```
{% if is_granted('ROLE_GESTIONNAIRE') %}
```

Pour y avoir accès seulement si on est ADMIN :

```
{% if is_granted('ROLE_ADMIN') %}
```

Evil Storie 2 :

En tant que personne malveillant et ayant découvert que la base de donnée est hébergée et pouvant être accéder par phpmyadmin, je vais donc effectuer un brute force (avec une wordlist) et comme utilisateurs par défaut ADMIN pour pouvoir essayer d'entré et avoir accès a toute la base de donnée mysql

En tant que développeur je vais donc changer l'utilisateur pour éviter qu'il s'appelle admin comme par default et aussi complexifié le mot de passe (je n'ai actuellement pas phpmyadmin sur mon ordinateur mais bien entendu si j'upload mon site je vais pouvoir passé par un service tiers qui lui proposera phpmyadmin souvent par exemple : www.vetux-line.com/phpmyadmin), je vais sécuriser tout cela sauf que moi je passerais par mon serveur Linux.

J'ouvre le terminal et je tape :

mysql -u (USER) -p (PASSWORD QUE L'ON TAPERA JUSTE APRÈS)

-> Donc par default sa sera mysql -u root -p (rien)

-> Sauf que j'ai déjà modifier tous ça mais nous allons le refaire ensemble avec un nouvelle user qu'on appeler « needsecu »

```
enzo@MacBook-Pro-de-Enzo:~$ mysql -u Vetux -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 46
Server version: 8.0.26 Homebrew

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Il faudra donc un mot de passe assez compliquer a trouver je vous conseille d'en générer un et de le garder de côté (de manière sécuriser) (MDP GEN :

« oq^Nq^K\$RSfn »)

-> CREATE USER 'needsecu@'localhost' IDENTIFIED BY 'oq^Nq^K\$RSfn';

Pour rappel si on ne se souviens pas du nom de la DATABASES on peut donc effectuer cette commande :

-> SHOW DATABASES;

-> GRANT ALL PRIVILEGES ON vetuxline . * TO 'needsecu'@'localhost'; (on donne toute les permissions a needsecu sur la base de donnée vetuxline

-> FLUSH PRIVILEGES; (pour mettre a jours les droits)

Maintenant nous devons modifié cela au niveau du fichier .env :

```
DATABASE_URL="mysql://needsecu:oq^Nq^K$RSfn@127.0.0.1:3306/vetuxline?serverVersion=8.0.26"
```

Une fois cela fais nous avons sécuriser notre Base de donnée au niveau du bruteforce.

Evil Storie 3 :

En tant que personne malveillante j'ai découvert que Vetux-Line utilisais une ancienne version de Symfony pas a jour je vais donc me documenter sur les failles que l'ancienne version avais et l'exploité

En tant que développer je fais bien attention d'avoir a jours Symfony et tous les autres packets associé a mon projet

en exécutant cette commande : symfony self:update, je peux aussi faire une commande supplémentaire pour vérifier qu'il n'y a pas d'autre faille de sécurité en exécutant cette commande : symfony security:check pour vérifier que tous les packets que nous avons installé n'ont pas de faille détecter a ce jours.

Evil Storie 4 :

En tant que personne malveillante je vais essayer d'upload a la place d'un fichier .csv un Shell me permettant d'avoir accès a tout le projet et les différents fichiers ou un fichier qui fera planté le système.

En tant que développer je dois donc vérifier que mon upload a seulement la possibilité d'upload un type de content, c'est a dire seulement les fichiers .csv et rien d'autre.

```
$filetype = $file->getMimeType();
```

```
if (str_contains($filetype, '/csv')){
```

```
} else {
```

```
DO NOTHING  
}
```

Evil Storie 5 :

Il y a eu une fuite d'information comme Twitch par exemple, il y avait tout le code source du projet mais aussi toute la database en fichiers .sql, en tant que personne malveillante je vais donc lire ce fichier .sql et essayer d'utiliser les mots de passe et users sur d'autre site pourquoi pas ou sur Vetux-Line

En tant que développeur j'ai au préalable sécurisé la base de donnée pour si jamais une grosse fuite arriverait à se produire en cryptant tout les mots de passe pour éviter qu'une personne ayant accès à la base de donnée ou via une fuite puisse réutiliser les mots de passe des utilisateurs car plusieurs études (<https://www.cyclonis.com/fr/rapport-83-pour-cent-utilisateurs-interrogues-utilisent-meme-mot-de-passe-plusieurs-sites/> ou <https://www.zdnet.fr/actualites/mot-de-passe-1-personne-sur-7-n-en-utilise-qu-un-seul-partout-39829024.htm> et bien d'autre..) démontre que les personnes utilisent souvent le même mot de passe un peu partout.

J'ai donc crypté le mot de passe et pour cela j'ai donc importé un module permettant de le faire :

```
use  
Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;
```

Et j'ai configuré le hashage du password :

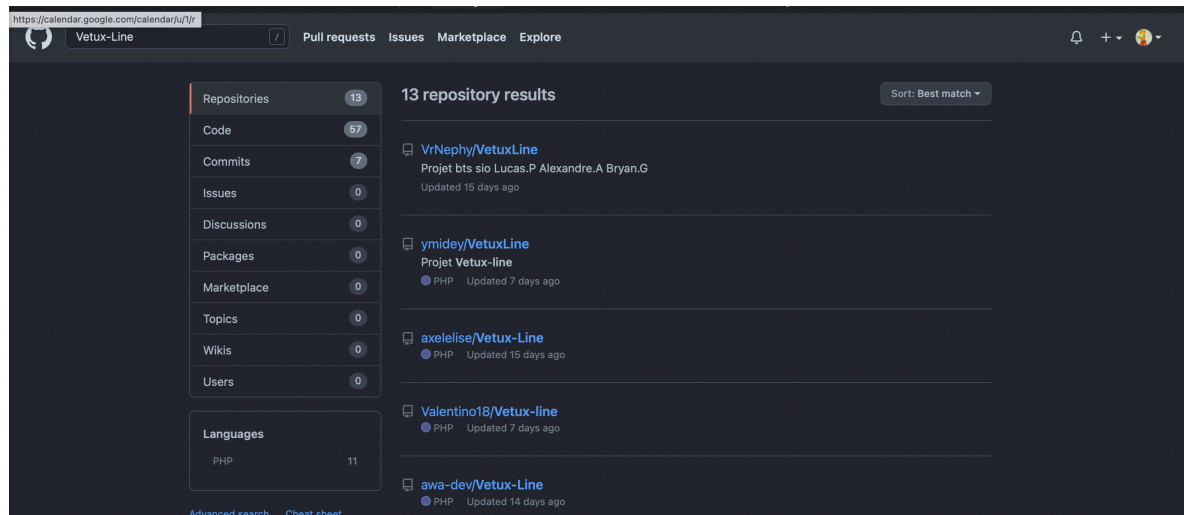
```
$user->setPassword(  
    $userPasswordHasherInterface->hashPassword(  
        $user,  
        $form->get('plainPassword')->getData()  
    )  
);
```

Comme ça lors de l'inscription le mot de passe sera crypté dans la base de donnée et si jamais elle fuit la personne ne pourra pas avoir accès au mot de passe.

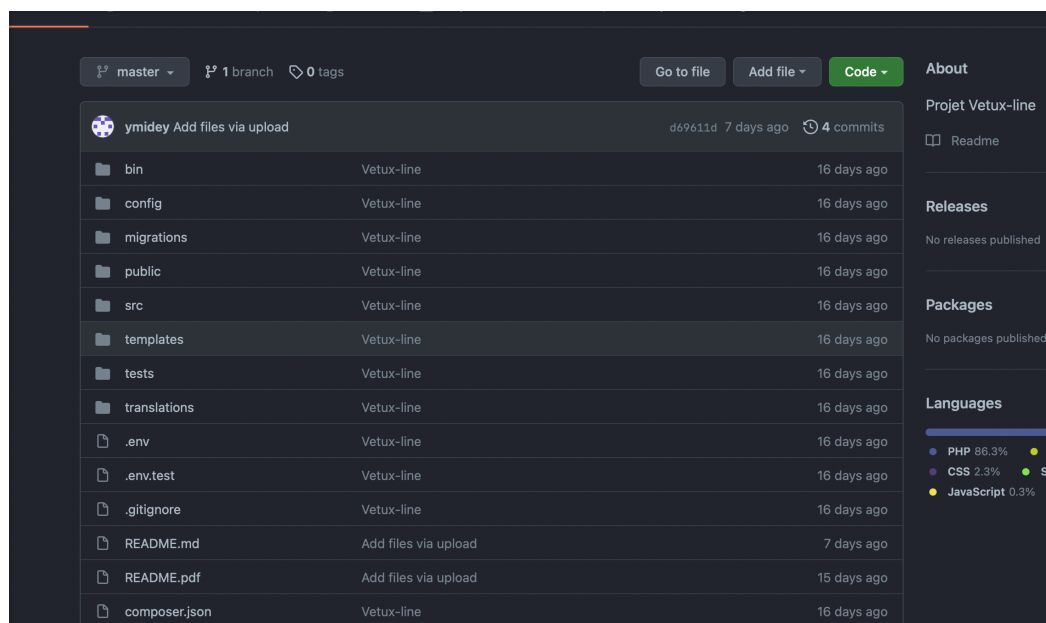
Evil Storie 6 :

En tant que personne malveillante je souhaiterais chercher via le nom du projet sur une base open source de stockage comme Github, Gitlab.. alors je cherche dans la barre de recherche tout simplement le nom du projet.

En tant que moi je vais tester cette théorie recherchons par exemple sur github.com le nom du projet Vetux-Line :



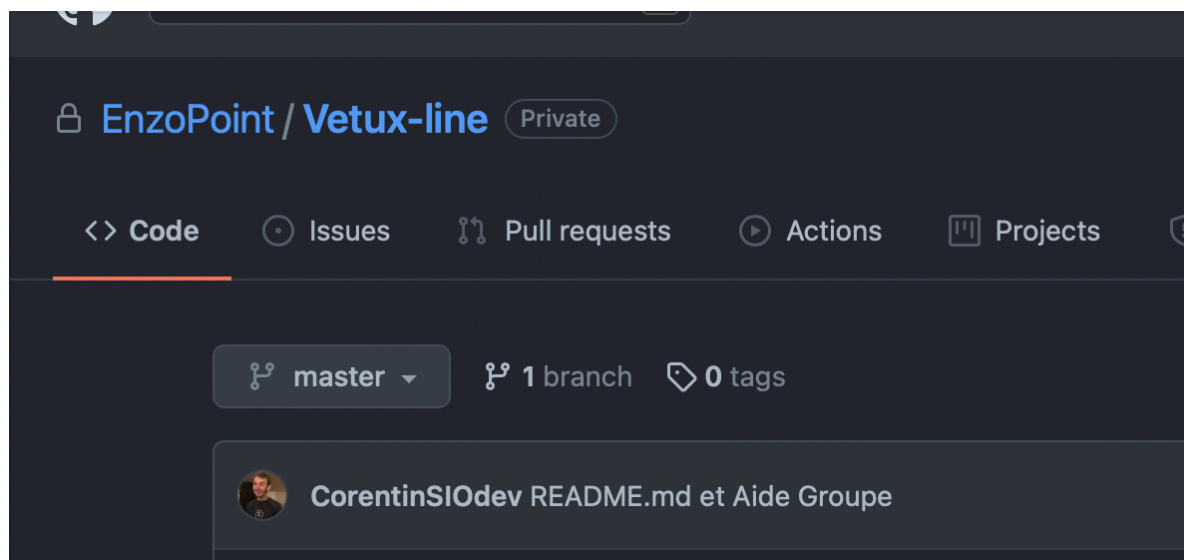
13 correspondance, entrons sur le premier par exemple ou plutôt le deuxième BOOM :



Nous avons accès au fichier .env et d'autre fichier sensible voyons ce qu'il y a dans le fichier .env

Nous avons donc le mot de passe de la base de donnée Mysql....
 DATABASE_URL="mysql://Vetuxline:sio@127.0.0.1:3306/vetuxline?
 serverVersion=mariadb-10.4.21"

En tant que développeur je veux éviter cela lorsque j'upload mon projet je vais donc le basculer en privé avec accès sur invitation (seulement les personnes invités au projet auront accès).



Le projet est donc sécurisé.