

Matière : MENACES, VULNÉRABILITÉS DES  
APPLICATIONS WEB

Date : 21/03/2022

Formateur : Malerba Sylvain

Auteur : Malerba Sylvain

# OWASP

# 0

# Liste des failles

- 1 - Injection
- 2 - Piratage de session
- 3 - Exposition de données sensibles
- 4 - Entités externes XML (XXE)
- 5 - Contournement du contrôle d'accès.
- 6 - Security Misconfiguration
- 7 - Cross-Site Scripting (XSS)
- 8 - Désérialisation non sécurisée (Insecure Deserialisation)
- 9 - Utilisation de composants présentant des vulnérabilités connues
- 10 - Manque de surveillance et de monitoring

# 01

# Injection SQL

```
<?php

$name = "pomme";

$requete = "SELECT * FROM `products` WHERE `product_name` LIKE '" . $name . "'";

$result = mysqli_query($GLOBALS['DB'], $requete) or die('Erreur SQL !<br>' . $
    requete . '<br>' . mysqli_error());
$return = array();

if ($data = mysqli_fetch_array($result)) {

    $return = $data;

}

return $return;
```

# 01

# Injection SQL

```
$name = "1' or '1' = '1'";  
$requete = "SELECT * FROM `products` WHERE `product_name` LIKE '$name'";  
  
$result = mysqli_query($GLOBALS['DB'], $requete) or die('Erreur SQL !<br>' . $  
    requete . '<br>' . mysqli_error());  
$return = array();  
  
if ($data = mysqli_fetch_array($result)) {  
    $return = $data;  
}  
  
return $return;
```

# 01

# Injection SQL

```
$name = "pomme"; DROP TABLE 'products';

$requete = "SELECT * FROM `products` WHERE `product_name` LIKE '" . $name . "'";

$result = mysqli_query($GLOBALS['DB'], $requete) or die('Erreur SQL !<br>' . $
    requete . '<br>' . mysqli_error());
$return = array();

if ($data = mysqli_fetch_array($result)) {

    $return = $data;

}

return $return;
```

## • Les solutions:

- Validation des entrées
- ORM
- Conversion en chaînes de caractères

## Validation des entrées

- Contrôle côté client
- Contrôle côté serveur

# 01

# Injection SQL

## ORM

```
use Tutor\Entity\User;

$userRepo = $entityManager->getRepository(User::class);

$queryBuilder = $entityManager->createQueryBuilder();

$queryBuilder->select('u')
    ->from(User::class, 'u')
    ->where('u.firstname = :firstname')
    ->setParameter('firstname', 'Sylvain');

$query = $queryBuilder->getQuery();

echo $query->getDQL(), "\n";
```



# 01

# Injection SQL

## ORM

```
$queryBuilder = $entityManager->createQueryBuilder();  
$queryBuilder->delete(User::class, 'u')  
    ->where('u.id = :id')  
    ->setParameter('id', 5);  
  
$query = $queryBuilder->getQuery();  
  
echo $query->getDQL(), "\n";  
echo $query->execute();
```

# 01

# Injection SQL

## Conversion en chaînes de caractères

<https://www.php.net/manual/fr/mysqli.real-escape-string.php>

```
$requete = "SELECT * FROM `products` WHERE `product_name` = '" .  
    mysqli_real_escape_string($GLOBALS['DB'],$name) . "'";  
    $result = mysqli_query($GLOBALS['DB'], $requete) or die('Erreur SQL  
        !<br>' . $requete . '<br>' . mysqli_error());  
$return = array();  
  
if ($data = mysqli_fetch_array($result)) {  
    $return = $data;  
}  
return $return;
```

01

# Injection SQL

EXERCICE

# Piratage de session

- **Coté serveur :**
  - Session
- **Coté client:**
  - Cookie de suivi
  - Cookie de session

- **Que faire côté serveur?**

- ID en session interdit
- Gérer la durée de vie d'une session
  - Configuration serveur
  - Code PHP

# 02

## Piratage de session

- **Pour les cookies de suivi?**
  - **Impossibilité de pirater une session**
  - **Respect de la vie privée**

- **Pour les cookies de session?**
  - Vérifier que les cookies sont chiffrés lors de la transmission via HTTPS ;
  - Ne pas stocker d'informations d'identification en texte clair dans les cookies ;
  - Date d'expiration des cookies-session.

## Quelques conseils:

- Demander un mot de passe fort
- Changer de mot de passe régulièrement (credential stuffing)
- Eviter les attaques de force brute (blocage temporaire d'un compte)
- Pas de comptes par défaut .
- Double authentification.



**02**

# **Piratage de session**

## **Exercice**

03

# Protection des données en transit

Man in the middle



# Protection des données en transit

Site en HTTP = données qui transitent en clair

Solution => HTTPS

Données GET et POST

# Protection des données en transit

## Comment passer un site en HTTPS:

Exemple en Node.js

```
var https = require('https');  
  
https.createServer(function (req, res) {  
  
}).listen(8080);
```

# Protection des données en transit

## Comment passer un site en HTTPS:

Configuration serveur

<https://www.digitalocean.com/community/tu-secure-apache-with-let-s-encrypt-on-ubuntu>



# Protection des données en transit

## Données GET

- Récupération des données par URL
- Possibilité de modification de l'URL

# Protection des données en transit

## Données POST

- Données transitent par requête asynchrone par exemple
- Données facilement piratables
- Cross-Origin Resource Sharing (CORS).

# Protection des données en transit

## Conseil

- GET pour la récupération d'informations.
- POST pour les informations qui seront manipulées (modification ,ajout etc...)
- Les requêtes POST doivent utiliser HTTPS/SSL.
- Vérifier **Cross-Origin Resource Sharing (CORS)**. (HTTPS)



# Protégez les données stockées sur une application

- <https://www.php.net/manual/fr/function.hash.php>
- md5
- sha1 (256, 512)
- aes-256-cbc

# Protégez les données stockées sur une application

```
function encrypt($data){
    $key = base64_decode("4URf254tvU4DZc3j2X36cvu6vTUFBkb438L7pZPb");
    $encryption_key = base64_decode($key);
    // Generate an initialization vector
    $iv = openssl_random_pseudo_bytes(openssl_cipher_iv_length('aes-256-cbc'));
    // Encrypt the data using AES 256 encryption in CBC mode using our
    // encryption key and initialization vector.
    $encrypted = openssl_encrypt($data, 'aes-256-cbc', $encryption_key, 0, $iv)
    ;
    // The $iv is just as important as the key for decrypting, so save it with
    // our encrypted data using a unique separator (:)
    return base64_encode($encrypted . '::' . $iv);
}

function decrypt($data){
    $key = base64_decode("4URf254tvU4DZc3j2X36cvu6vTUFBkb438L7pZPb");
    $encryption_key = base64_decode($key);
    // To decrypt, split the encrypted data from our IV - our unique separator
    // used was ":"
    list($encrypted_data, $iv) = explode(':', base64_decode($data), 2);
    $result = openssl_decrypt($encrypted_data, 'aes-256-cbc', $encryption_key,
        0, $iv);
    return $result;
}
```

# Protégez les données stockées sur une application

## Coté BDD:

- Sécurisation de l'accès à la BDD
- Rendre les données sensibles non lisibles
- Masquer les données en BDD

04

# Protégez les données stockées sur une application

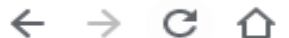
## Exercice

# Gestion des contrôles d'accès

- Contrôle des modifications URL
- Fichier .htaccess
- Accès fichiers utilisateurs
- LDAP et Active Directory

# Gestion des contrôles d'accès

## Contrôle des modifications URL

 [toto.fr/fr/lieu/fesch1/](http://toto.fr/fr/lieu/fesch1/) <https://toto.fr/user=3453>

# Gestion des contrôles d'accès

## Fichier .htaccess

```
RewriteBase /  
  
RewriteRule ^/?$ /Controller/VitrineController.php [L,NC]  
  
RewriteRule ^default/?$ /Controller/VitrineController.php [L,NC]  
  
RewriteRule ^admin/?$ /Controller/LoginController.php [L,NC]  
  
RewriteRule ^admin/create/?$ /Controller/AdmininistrationController.php [L,NC]  
  
RewriteRule ^admin/update/?$ | /Controller/AdmininistrationController.php [L,NC]
```

# Gestion des contrôles d'accès

Fichier .htaccess

Deny from all



05

# Gestion des contrôles d'accès

**Accès fichiers utilisateurs**

Question?

# Gestion des contrôles d'accès

## Accès fichiers utilisateurs

- Contrôle des accès aux fichiers
- Encoder le contenu des fichiers
- Hash par utilisateur

# Gestion des contrôles d'accès

## Résumé/ Conseil

- Éviter les sens logique dans l'URL des pages Web.
- Modifiez les noms par défaut de vos pages web.
- Contrôle d'authentification par page si nécessaire.
- Exception et code d'erreur

**05**

# **Gestion des contrôles d'accès**

## **Exercice**

# Le cross-site scripting (XSS)

**Les attaques cross-site scripting ou XSS** sont faites pour prendre le contrôle de du navigateur.

- **Accès aux données sensibles**
- **Modifications non autorisées à une application web**

# Le cross-site scripting (XSS)

## Conseils:

- **Appliquer la validation des données d'entrée** : Interdire certains caractères (balise script, commande UNIX)
- **Appliquer la transformation des entrées** : Encoder les données au format html par exemple (htmlentities).
- Configurer vos cookies avec le flag **HttpOnly**.

# Le Cross Site Request Forgery (CSRF)

## Conseils:

- Exiger la réauthentification pour toutes les demandes des utilisateurs.
- Utiliser un jeton unique pour chaque demande.
- Vérification du jeton côté client par rapport au jeton côté serveur.
- Utilisation de Bibliothèque CSRF

06

# Le Cross Site Request Forgery (CSRF)

## Exercice



Entité interne / Entité externe

```
<!ENTITY website "La cuisine de Sylvain">  
  
<!ENTITY profilPicture "No picture">  
  
<character>&website;&profilPicture;</character>  
  
  
  
<!ENTITY website SYSTEM "http://www.toto.fr">  
  
<!ENTITY profilPicture SYSTEM "file:///usr/picture">  
  
<character>&website;&profilPicture;</character>
```