Simulation de failles

18/09/2017 - M. HAMMOUDI

Céline GRUNENBERGER - Michael FROELICH

Master 2 MIAGE

32%

Des applications testées sont vulnérables aux attaques par injection SQL

6,6 Millions

De mots de passe volés à LinkedIn en Juin 2012

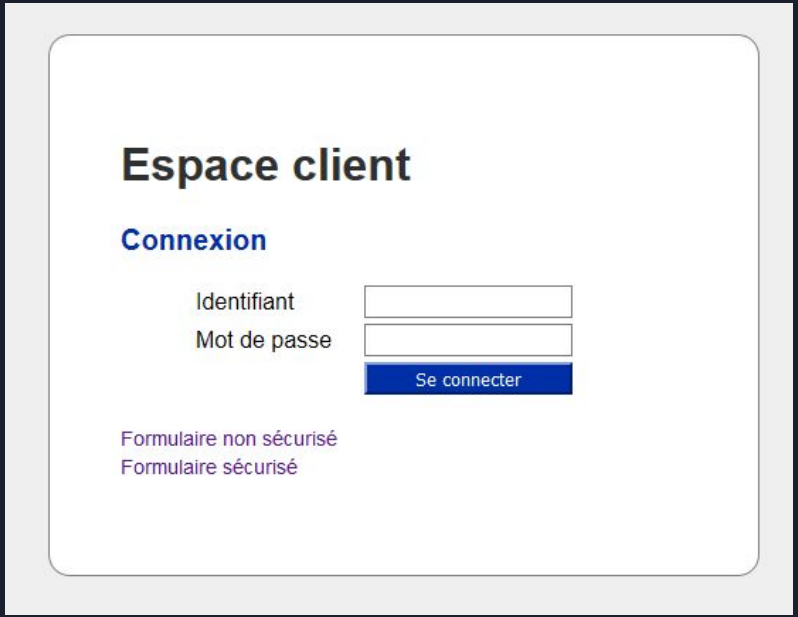
11 Millions

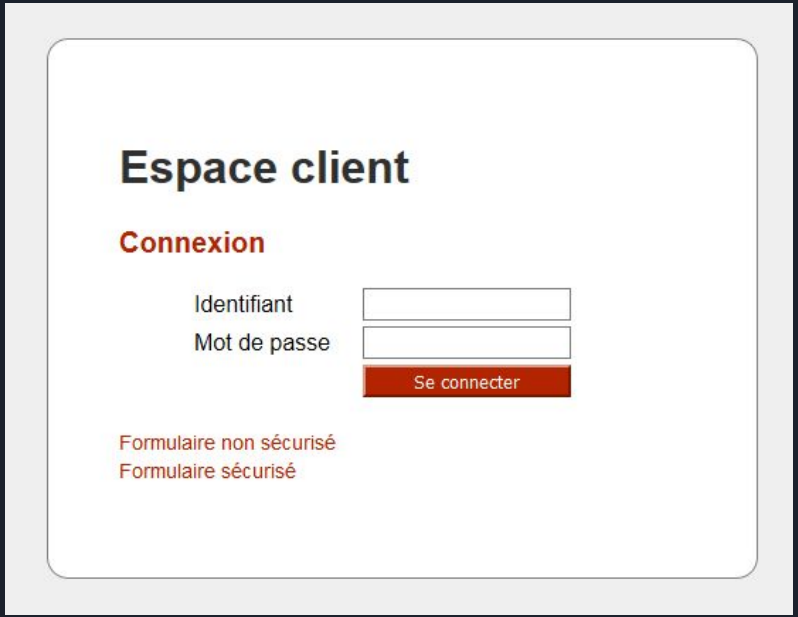
De mots de passe hashés + 8,2 Millions d’adresses e-mail volées chez Gamingon en Juillet 2012

Les failles testées

- ➔ Injection de commentaire SQL
- ➔ Condition OR 1 = 1
- ➔ Exécution de code JavaScript (XSS)
- ➔ UNION
- ➔ DROP TABLE

Présentation de l’application

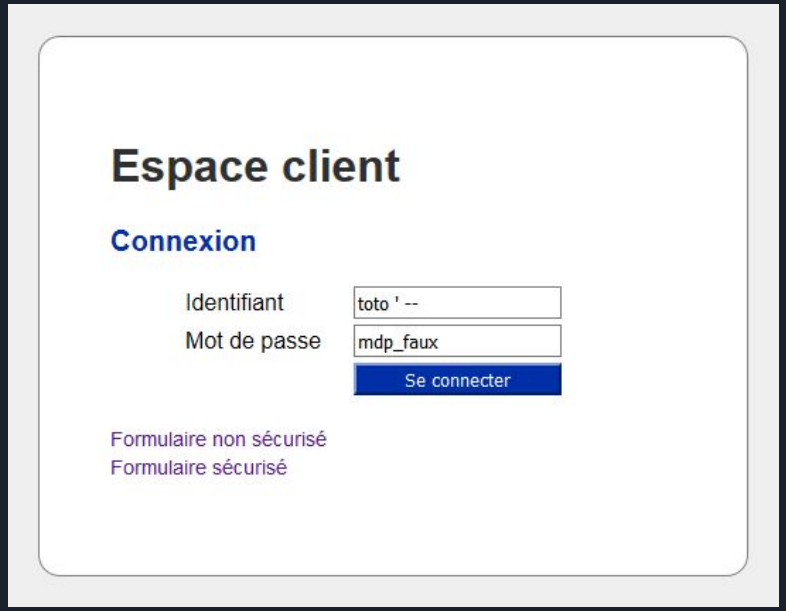


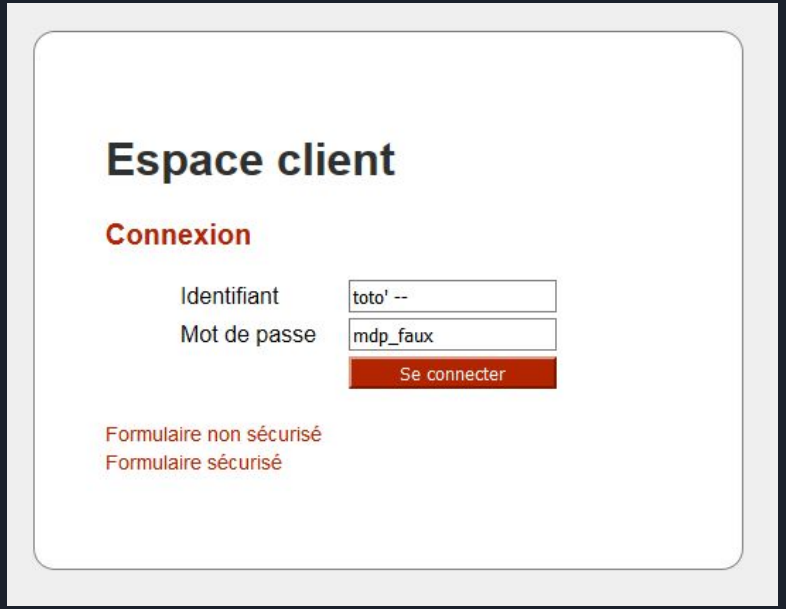


Présentation de l’application

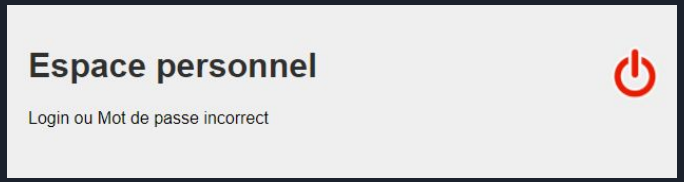
Injection de commentaire SQL

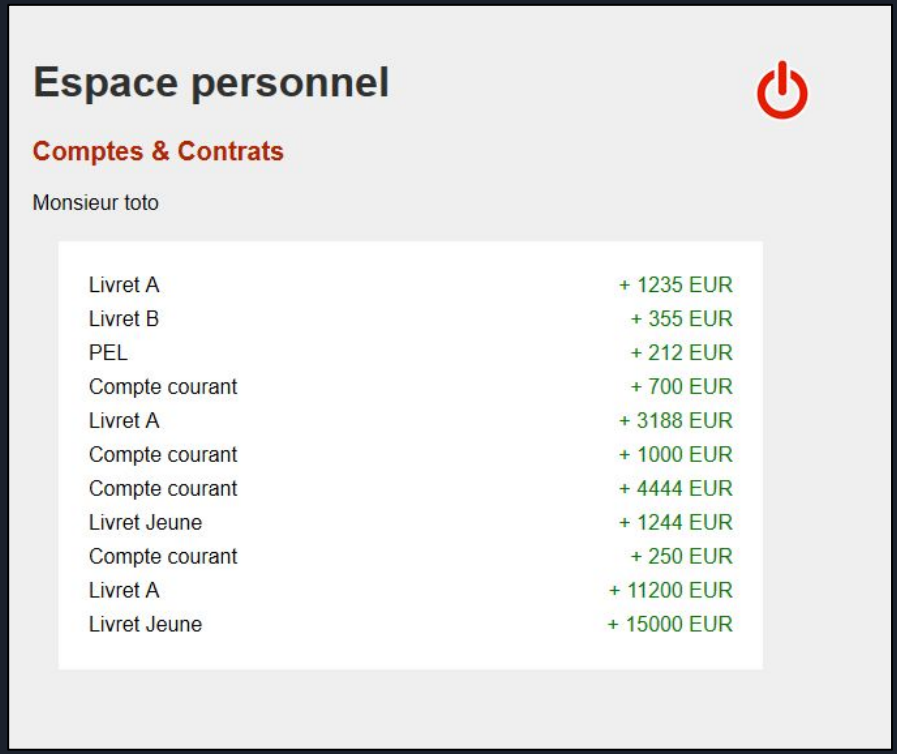
Injection d’un commentaire SQL





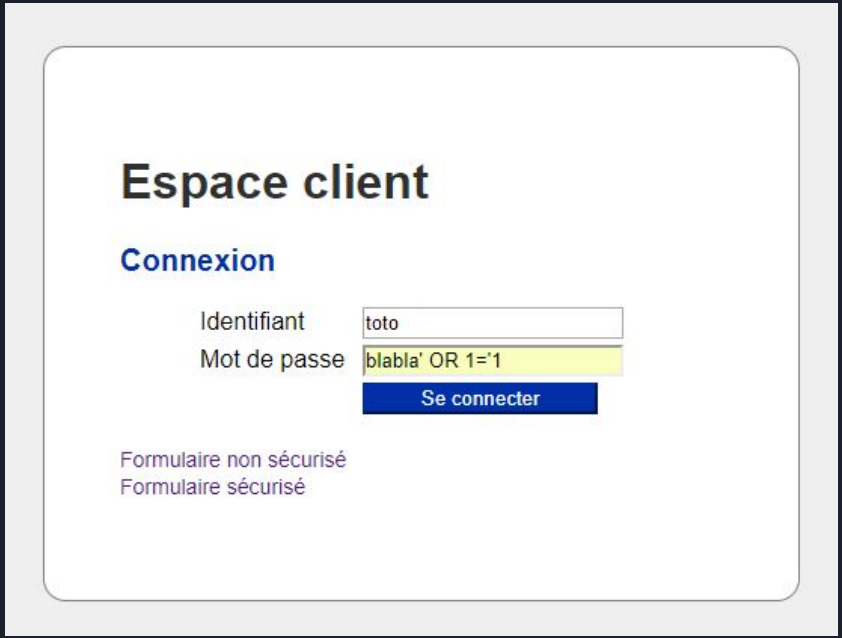
Injection d’un commentaire SQL

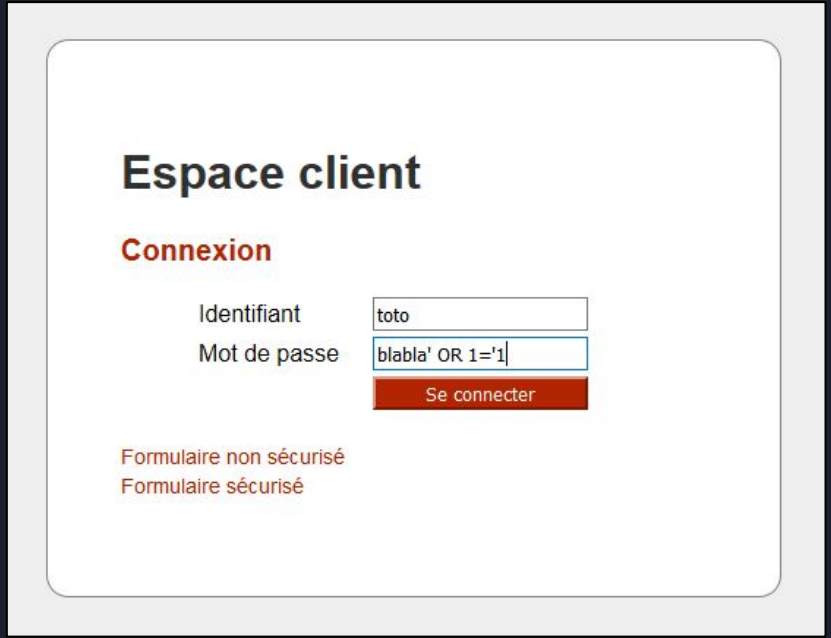




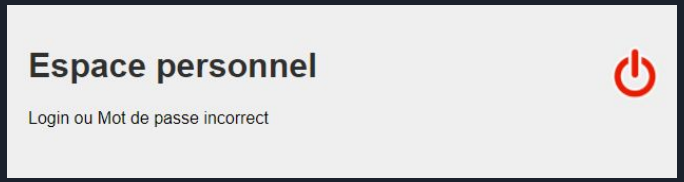
Injection de condition toujours vraie OR 1=1

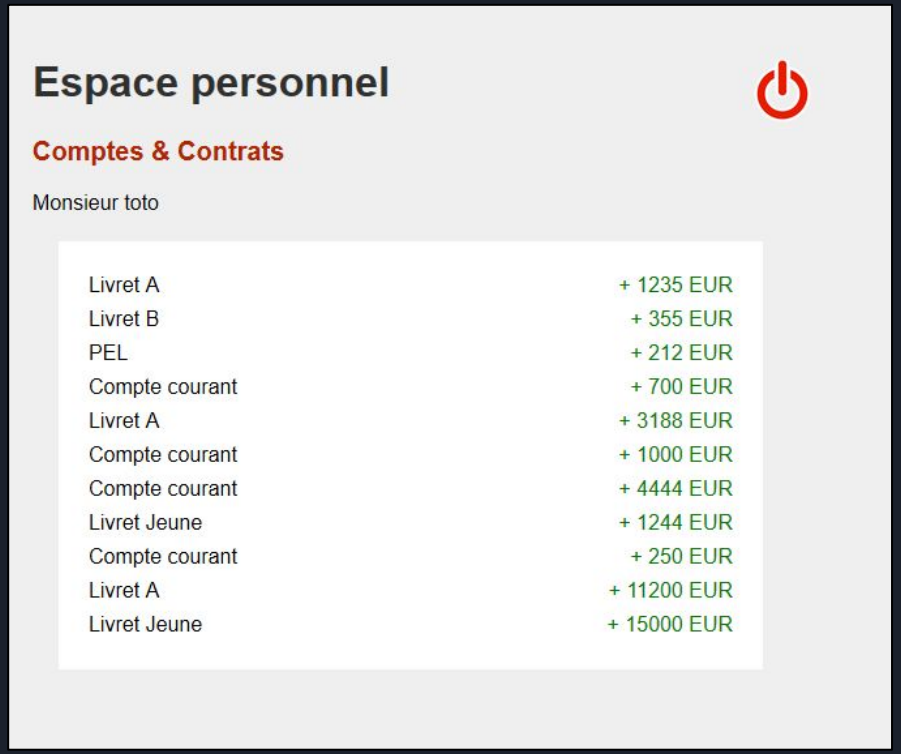
Injection de condition toujours vraie OR 1=1





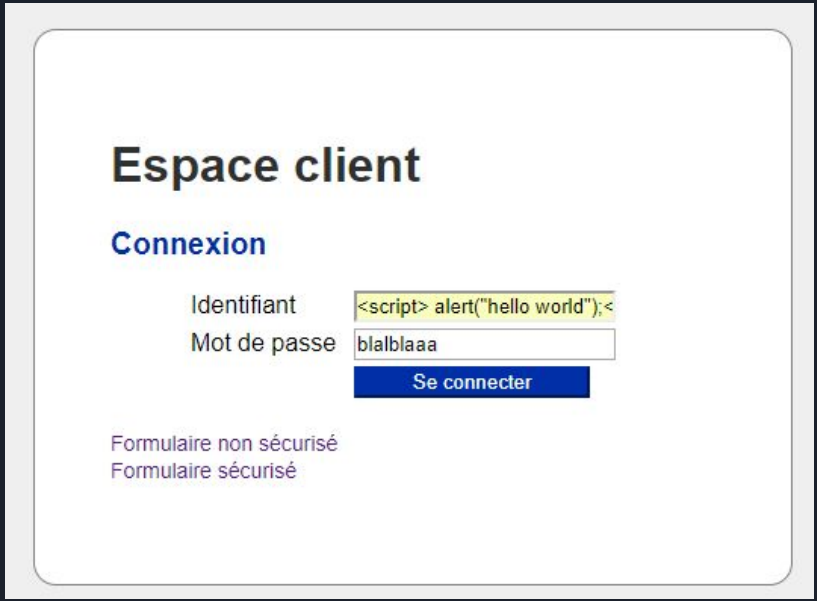
Injection de condition toujours vraie OR 1=1

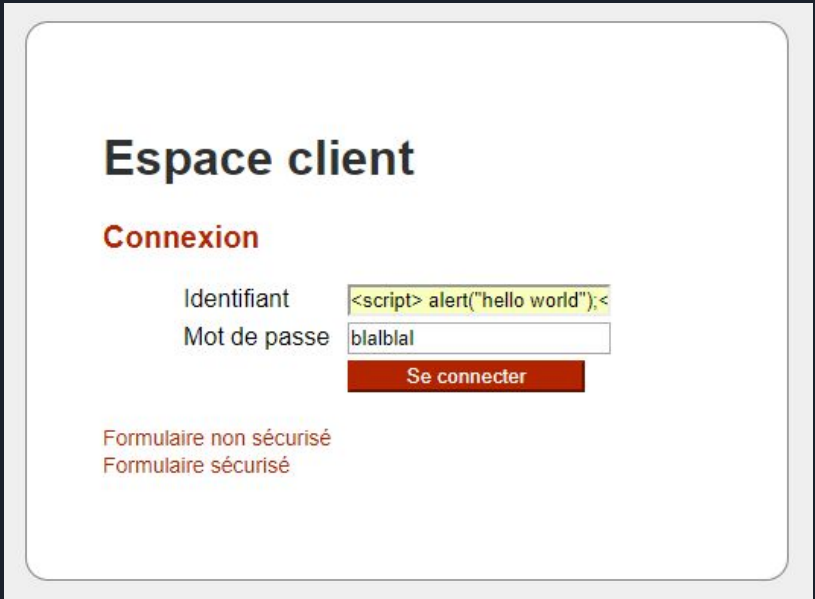




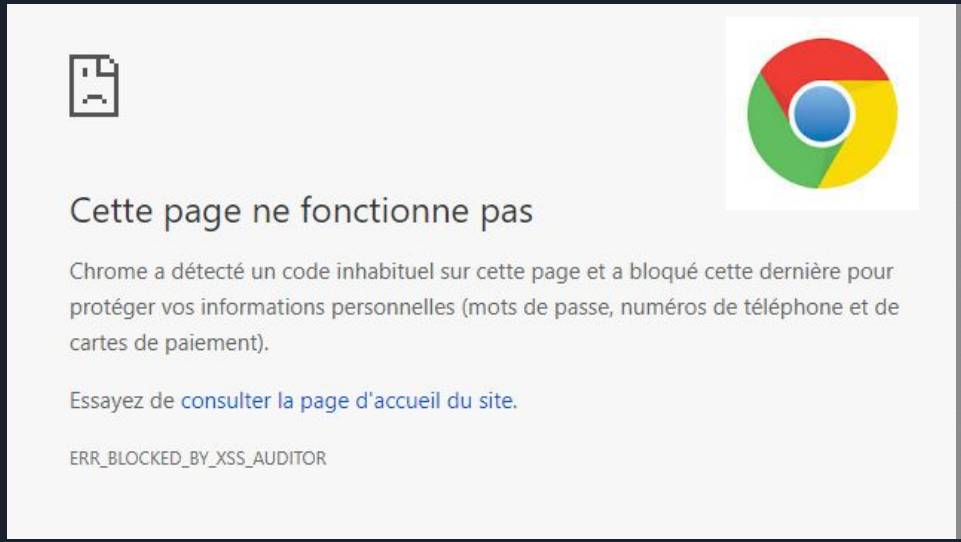
Exécution de code Javascript : XSS

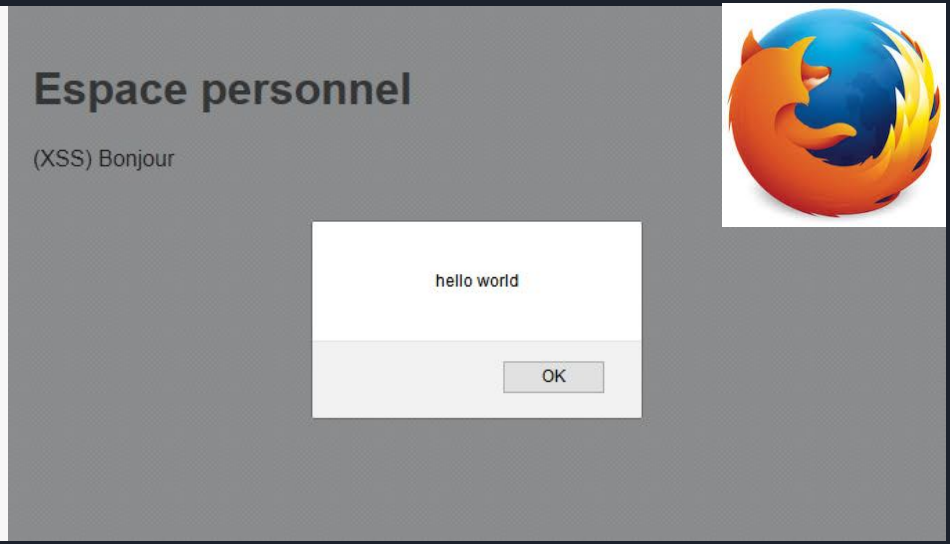
Exécution de code Javascript : XSS





Exécution de code Javascript : XSS

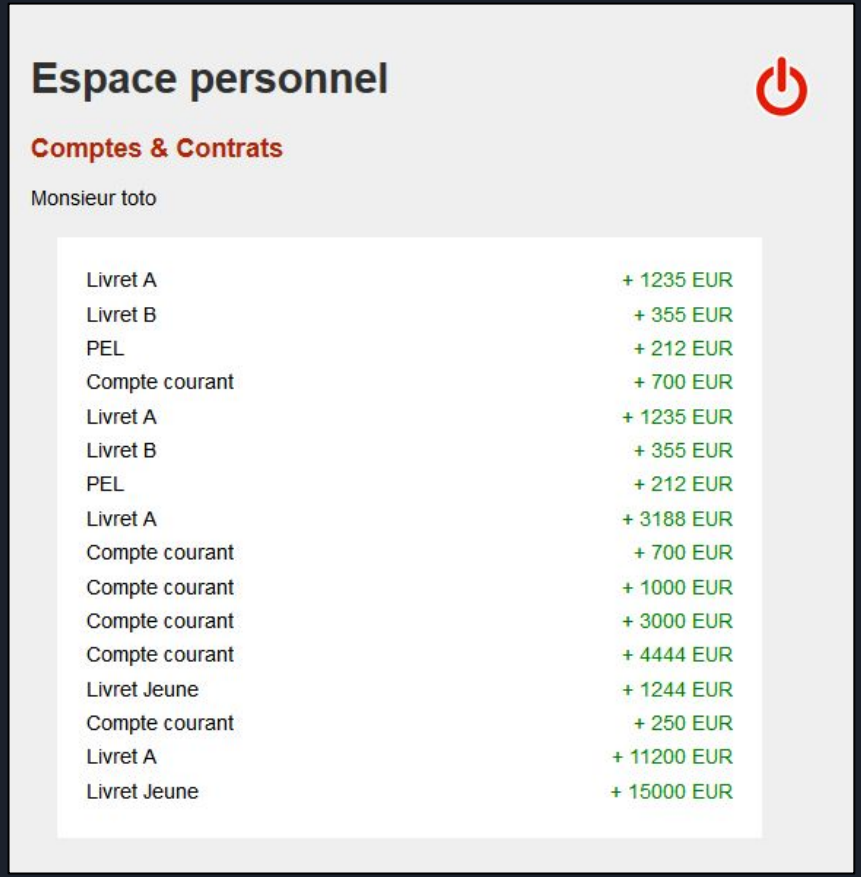




Afficher d’autres tables avec UNION

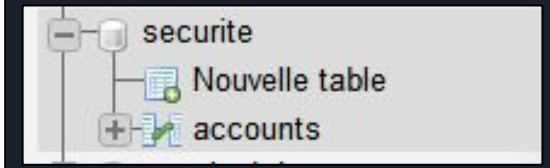
UNION

toto' UNION SELECT * FROM accounts --



Supprimer une table : DROP TABLE

Supprimer une table : DROP TABLE



Contrer les failles

Se protéger contre les CSRF (Cross site request forgery)

Transmission à un utilisateur connecté d'une requête falsifiée pointant vers une action interne au site.

$token = random_bytes();

→ Création d'une session avec un token

```php
<?php

session_start();
$token = bin2hex(mcrypt_create_iv(32, MCRYPT_DEV_URANDOM));
$_SESSION['token'] = $token;
?>
```

```php
<input type="hidden" name="token" id="token" value="<?php echo $token; ?>" />
```

```php
session_start();
// on vérifie que tous les jetons sont présents
if(isset($_SESSION['token']) AND isset($_POST['token']) AND !empty($_SESSION['token']) AND !empty($_POST['token']))
{
    // On vérifie que les deux correspondent
    if($_SESSION['token'] == $_POST['token'])
```

Ralentir les attaques par force brute ou par dictionnaire

Bombardement du formulaire avec des mots de passe générés ou couplés à une base de données de mots de passe fréquents.

→ Mise en place de cookies limitant le nombre de tentative de connexion

```php
session_start();

if(isset($_SESSION['nombre']) and $_SESSION['timestamp_limite'] < time())
{
    // Destruction des variables de session
    unset($_SESSION['nombre']);
    unset($_SESSION['timestamp_limite']);
}

// Si le cookie n'existe pas
if(!isset($_COOKIE['marqueur']))
{
    // Si le formulaire est rempli
    if(isset($_POST['connexion']) AND !empty($_POST['login']) AND !empty($_POST['mot_de_passe']))
    {
        // Initialisation du compteur
        $_SESSION['nombre'] = 0;
        $_SESSION['timestamp_limite'] = time() + 60;
    }
```

```php
// Si le cookie marqueur n'existe pas on le crée
if(!isset($_COOKIE['marqueur']))
{
    $timestamp_marque = time() + 60; // On le marque pendant une minute
    $cookie_vie = time() + 60*60*24; // Durée de vie de 24 heures pour le décalage horaire
    setcookie("marqueur", $timestamp_marque, $cookie_vie);
}
```

Empêcher les injections SQL

- session_start();
- isset();
- htmlspecialchars();
- Requêtes préparées : **$req = $bdd->prepare("SELECT * FROM users WHERE users.login=? AND users.password=?");**

Bilan et best practices

- Hasher les mots de passes dans la BDD
- Utiliser des requêtes préparées
- Ne jamais faire confiance aux données reçues

Merci pour votre attention

UNIVERSITÉ HAUTE-ALSACE

miage Mulhouse

18/09/2017 - M. HAMMOUDI

Céline GRUNENBERGER - Michael FROELICH

Master 2 MIAGE

Sources & applications

- Git & GitHub
- OpenClassroom
- Developpez.net
- Cours