Sécurisez vos applications

Initiation

TABLE DES MATIERES

Section 1 But de cette formation

Nous allons parcourir les éléments qui nous permettront de

- Identifier les vulnérabilités d'une application web ;
- Assurer la sécurité des identités des utilisateurs d'une application ;
- Implémenter les outils de protection des données d'une application ;
- Appliquez les principes de sécurité dès la conception d'une application.

Sites web utilisés :

- https://openclassrooms.com/fr/courses/1761931-securisez-vos-applications
- https://www.cloudflare.com/fr-fr/learning/security/threats/owasp-top-10/
- site officiel OWASP https://owasp.org/www-project-mobile-top-10/2023-risks/

Section 2 Des exemples de vulnérabilités

Les exemples sont malheureusement nombreux :

Il suffit de taper « faille de sécurité » pour voir l'étendue ...

Ransomweare sur les hôpitaux

Attaque par les russes sur les réseaux de transport et de défense Ukrainiens.

Influence/modification de l'opinion publique américaine par des hackers du gouvernement russe.

D'autres exemples ici :

https://openclassrooms.com/fr/courses/1761931-securisez-vos-applications/5702477-apprivoisez-les-vulnerabilites-des-applications-et-leurs-consequences

Ce site a montré des exemples sur :

- la **sécurité des accès**
- failles dans l'authentification et les autorisations aux applications
- le **canal de communication qui n'est pas sûr** dans des transmissions RFID
- **la mauvaise manipulation de données non fiables**

Au sens large, la sécurité c'est l'ensemble des moyens (techniques, organisationnels, humains, légaux) pour **minimiser la surface d'exposition d'une application ou d'un système contre les menaces** .

Menaces :

- **Passives** visant à écouter ou copier des informations illégalement ;
- **Actives** consistant à altérer des informations ou le bon fonctionnement d'un service.

Section 3 Les 6 aspects de la sécurité d'une application

Comme pour le vol dans la vie courante, le risque 0 n'existe pas.

Les démarches consistent à le réduire.

Sécuriser une application, c'est **s'intéresser notamment aux accès ou bien aux données qui communiquent avec l'extérieur**.

La sécurisation d'une application ou d'un système s'attache aux 6 aspects suivants :

- **L'authentification** ;
- **Le contrôle d'accès** ;
- **L'intégrité** des données ;
- **La confidentialité** des données ;
- **La non-répudiation** ;
- La protection contre **l'analyse du trafic**.

3.1 L'authentification

L'authentification sert à savoir QUI tente d'utiliser l'application.

QUI peut désigner une personne ou un autre système (cas d'une API).

Par exemple pour un distributeur de billets, l'authentification se fait par la carte bancaire et le code à 4 chiffres.

Sur son PC il faut un code pour le déverrouiller.

Les types d'authentification peuvent être :

- par identifiant et mot de passe
- par adresse mail et mot de passe
- par un certificat
- par carte
- par plusieurs de ces modes (ex carte et code) : authentification multimodale

Dans ce cas les identifiants, codes et mots de passe doivent être sauvegardés de façon cryptée.

3.2 Le contrôle d'accès

Ce n'est pas parce que le QUI s'est correctement identifié qu'il a accès à toute l'application.

Le contrôle d'accès permet d'allouer des droits par catégories d'utilisateurs : certains seront administrateur, d'autres commerciaux, d'autres de simples acheteurs …

On réalise alors une application qui permet de gérer des **rôles** par catégories d'utilisateur.

A tout nouvel utilisateur est associé un rôle.

3.3 L'intégrité

Définition :

État de quelque chose qui a conservé sans altération ses qualités, son état originels : Conserver l'intégrité de ses facultés intellectuelles malgré l'âge.

Il s'agit ici de prévenir l'altération volontaire ou accidentelle d'une donnée ou des services d'un système. Elle s'applique à la phase de **communication** entre composants, au **flux,** au **stockage** des données (altérations de contenu) et au **système** (détection d'intrusion).

Des axes pour assurer l'intégrité sont le **chiffrement** des données et l'ajout de **signatures** numériques

3.4 La confidentialité des données

Les données sensibles sont considérées comme confidentielles et doivent être protégées (mot de passe, données bancaires ou médicales.)

Il s'agit de garantir que des données acquises illégalement soient inutilisables.

Les moyens technologiques principaux pour mettre la confidentialité en œuvre reposent sur des **mécanismes de chiffrement** qui permettent de protéger l'échange et le stockage des données.

3.5 La non répudiation

Cette fonction consiste à s'assurer que **l'envoi et la réception d'un message sont incontestables**.

En d'autres termes, l'émetteur ou le récepteur d'une donnée ne doit pas être en mesure de nier son implication en cas de litige. Le moyen technologique repose sur les certificats .

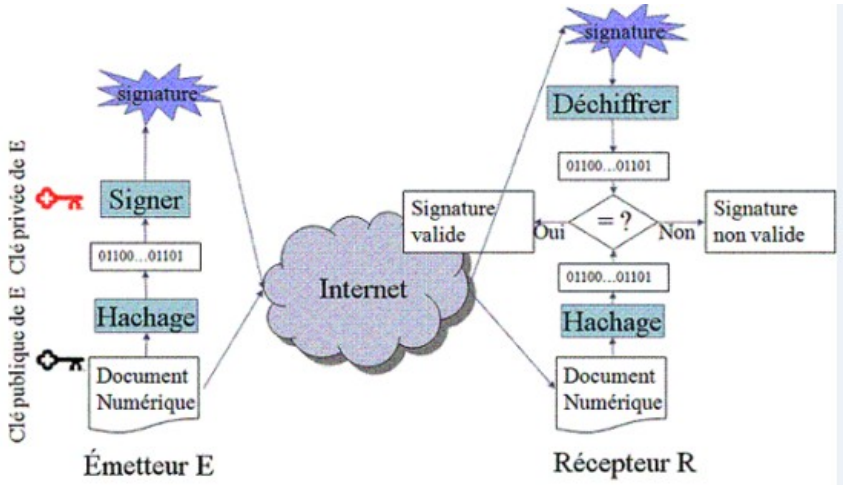
La signature digitale :

La signature digitale est un mécanisme cryptographique qui permet d'assurer la non répudiation de l'origine.

Ce mécanisme repose sur un système cryptographique asymétrique

La signature est calculée en utilisant la clé privé de l'émetteur

La signature est vérifiée en utilisant la clé publique de l'émetteur



(source https://moodle.utc.fr/pluginfile.php/16777/mod_resource/content/0/SupportIntroSecu/co/CoursSecurite_15.html)

L'émetteur du message génère sa paire de clés (publique, privée).

Il diffuse sa clé publique et garde sa clé privée secrète.

Pour signer un document l'émetteur commence par calculer le code hashage du document puis signe ce code de hashage avec sa clé privée.

Le résultat de cette dernière opération (chiffrement avec clé privée) est la **signature digitale** qui accompagnera le document.

Quand le récepteur reçoit le message et la signature digitale, il recalcule le code de hashage, déchiffre la signature avec la clé publique de l'émetteur et compare les deux codes de hashages. Si les deux codes sont similaires alors la signature est valide.

L'émetteur ne pourra pas nier dans le futur avoir émis le message puisque n'y a que lui qui peut générer la signature digitale avec sa clé privée secrète.

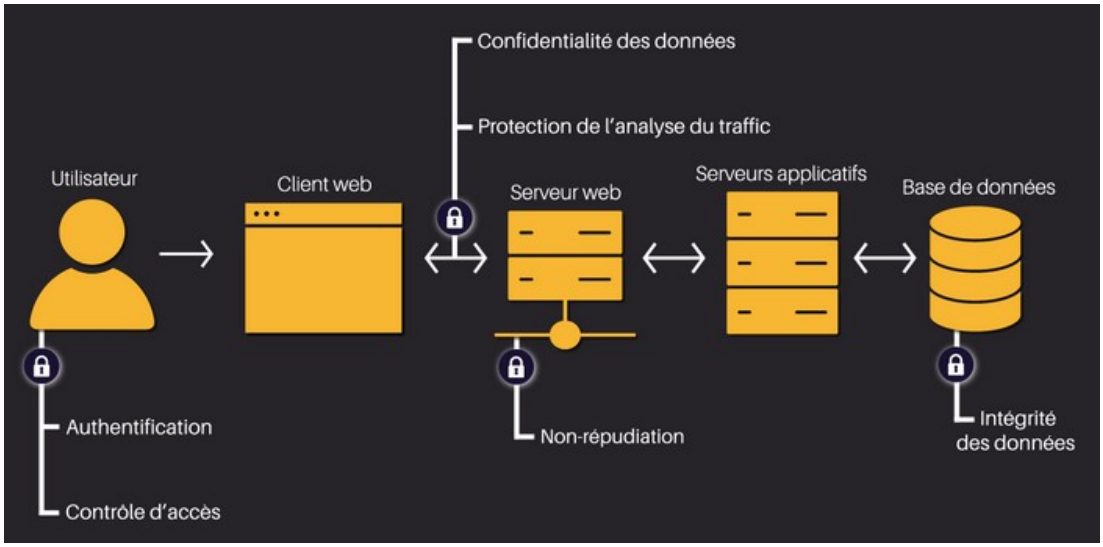
3.6 Protection contre l'analyse du trafic

Il faut faire en sorte que des données écoutées soient inutilisables.

Une solution est l'utilisation des protocoles **SSL** (Secure Socket Layer) et **TLS** (Transport Layer Security) dans les échanges sur le Web grâce au protocole **HTTPS** .

3.7 Résumé

(dessin openclassroom)



Section 4 Communications sécurisées avec TLS

4.1 Problématique

Un canal de communication non sécurisé entre deux équipements peut être intercepté par un troisième à des fins malveillantes : on parle d'une attaque de l'homme du milieu.

Imaginez que vous vous connectiez à un **routeur sans fil non fiable**, comme un accès Wi-Fi public gratuit.

Vous essayez alors d'**accéder à votre compte en banque** via le site de votre banque en ligne. Dans la plupart des cas, vous auriez obtenu une **erreur de certificat** vous indiquant que le site de votre banque ne possède pas les certificats appropriés.

En d'autres termes, **le navigateur vous prévient que le site de votre banque ne possède pas de « carte d'identité » valide** (faille sur la fonction authentification).

Vous ne faites pas attention (c'est les vacances) et vous acquittez le message d'erreur. **Vous vous connectez à votre compte avec vos identifiants**, vous réalisez quelques opérations : tout semble normal !

En réalité, une personne malveillante a pu mettre en place un serveur qui **ressemble à celui de votre banque**.

Quand vous vous y connectez, ce serveur rapatrie la page web de votre banque, la modifie un peu et vous la présente.

Lorsque vous vous connectez, ce serveur intermédiaire **enregistre les détails de votre compte**, se connecte à votre place, accède aux détails de votre compte et vous en envoie une copie.

Il s'agit d'une **faille sur la fonction confidentialité**.

4.2 Le protocole TLS

TLS (Transport Layer Security), successeur de **SSL** (Secure Socket Layer) est à employer.

TLS assure 3 fonctions majeures pour la sécurité des communications inter-applications : **la confidentialité, l'intégrité et l'authentification**.

Confidentialité :

Les données espionnées ne doivent pas être lues.

Cette fonctionnalité est assurée par **des algorithmes de chiffrement symétriques** pour l'échange des données (AES 256, RC4, DES, BlowFish) et **asymétriques** (RSA, DSA, Diffie Helmann) dans la phase d'authentification et d'échange de la clé secrète.

Intégrité :

Le client et le serveur doivent pouvoir **s'assurer que les messages transmis ne sont pas altérés** et qu'ils **proviennent bien de l'expéditeur attendu**. Dans le cas de TLS, ces fonctionnalités sont assurées par la signature des données (HMAC, MD5, RIPEMD, SHA-1)

Authentification :

Consiste a être sûr sur celui qui se connecte.

Cette fonctionnalité est implémentée grâce à l'utilisation de certificats X.509.

4.3 Mise en place de https en local

En utilisant l'application jointe, ouvrir la fenêtre de debug de votre navigateur par F12. Aller dans l'onglet Réseau

Inscrire un nouvel utilisateur et constater que les données sont en clair.

Le site https://www.youtube.com/watch?v=iamsyYFCH70 a inspiré ce qui suit.

La mise en place de https nécessite un certificat propre à la société qui produit le logiciel.

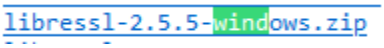
Ce certificat est payant.

Une autorité en propose depuis peu des gratuits : https://letsencrypt.org/fr/

Nous allons créer un certificat auto signé. Créer un certificat et une clé privée.

Aller sur le site suivant et télécharger libressl

https://ftp.openbsd.org/pub/OpenBSD/LibreSSL/



Décompresser le fichier et noter le chemin x64 où se trouve openssl.exe

Modifier la variable d'environnement PATH pour ajouter ce chemin.

Créer le répertoire c:\libressl

Créer le sous répertoire c:\libressl\ssl

Y placer le fichier openssl.cnf qui vient de apache\conf

Ouvrir une fenêtre cmd et se déplacer dans le répertoire c:\wamp64\apacheXX.conf

taper la commande :

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
macle.key -out macle.crt
```

Répondre aux questions comme ceci :

```
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:
```

Ceci doit créer le fichier macle.key et macle.crt

Dans apache/conf faire une copie de http.conf par sécurité.

Editer httpd.conf

Décommenter les 3 lignes contenant

- mod_ssl.so
- mod_socache_shmcb.so
- Include conf/extra/httpd-ssl.conf

Sauvegarder le fichier

faire une copie du fichier apache/conf/extra/httpd-ssl.conf

L'éditer

remplacer la ligne ServerName www.example.com:443

par

ServerName localhost

Plus loin mettre ceci : (lignes 145 et 155)

SSLCertificateFile "${SRVROOT}/conf/macle.crt"

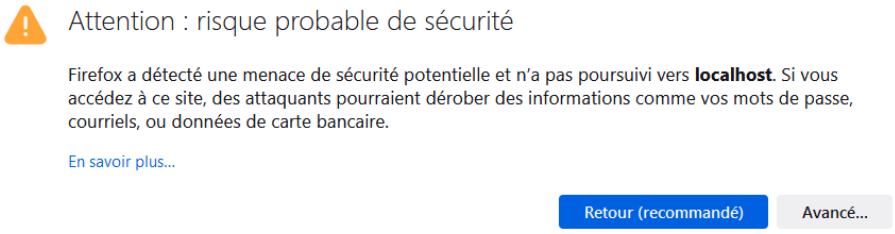
SSLCertificateKeyFile "${SRVROOT}/conf/macle.key"

Sauvegarder.

Relancer le serveur http

Demander la page d'accueil. Par https://localhost

On doit voir :



Ceci montre que https fonctionne mais l'on a pas payé une autorité de certification, le navigateur ne dispose pas de la clé pour être sûr de l'authentification du site.

Pour poursuivre dans ce mode appuyer sur

Avancé...

puis sur

Accepter le risque et poursuivre

Les versions actuelles des navigateurs ne permettent plus de poursuivre. Le contenu du site ne sera pas visible.

Le fait d'avoir autosigné ne garantit rien, tout individu malveillant peut faire de même. Le navigateur refuse l'accès

Mais si vous avez ce comportement signifie que la configuration avec certificat fonctionne.

Si on souhaite que notre certificat soit valide sur un vrai projet il faut l'obtenir depuis une autorité de certification

Pour une configuration sous Linux, utiliser la vidéo à la minute 28 :

https://www.youtube.com/watch?v=iamsyYFCH70

Section 5 Mettre en place une authentification

5.1 Problématique

L'authentification permet de savoir QUI se connecte à l'application.

Il faut que le moyen soit sûr sans être espionné.

Aujourd'hui, la plupart des applications utilisent encore le **mot de passe** comme moyen d'assurer l'authentification. Or, cette méthode doit être considérée comme peu sûre. En effet, il est assez simple de s'attaquer à ses vulnérabilités :

- Attaque facilitée par le caractère **prédictif** du mot de passe : certains de vos utilisateurs utilisent leur date de naissance, ou le nom de leur femme ou encore 123456 ;
- Attaque par **force brute** : par dictionnaire par exemple . Un programme client malveillant essaye toutes les combinaisons possibles. D'autant plus long que le mot de passe est long.
- Attaque sur des bases de données **contenant des mots de passe mal sécurisés** ;
- Attaque par **phishing (**hameçonnage). C'est par exemple le mail contenant une fausse information d'une banque qui demande des informations.

Exemple de spotify : https://openclassrooms.com/fr/courses/1761931-securisez-vos-applications/5702584-mettez-en-place-un-mecanisme-d-authentification#/id/r-5734043

5.2 Sécuriser le stockage des mots de passe

Le mot de passe ne doit pas être stocké en clair !

Il faut stocker un mot de passe crypté avec une fonction de hachage comme MD5 ou SHA256. SHA256 est maintenant préconisé par rapport à MD5.

L'utilisation d'une donnée « sel » permet de rendre plus difficile l'accès au mot de passe.

Voici du code php

```
<?php
  $options=array(
   'salt'=>random_bytes(22,MCRYPT_DEV_URANDOM),
   'cost'=>12,
 );
        $password_hash=password_hash($password_string,PASSWORD_BCRYPT,
$options);
?>
```

Dans l'application fournie, implémenter le cryptage du mot de passe.

Les méthodes impactées de la classe ManageUsers sont addUser et verifyUser

password_verify() teste si le mot de passe crypté correspond.

5.3 Authentification multifactorielle

Mettre en place une **authentification multifactorielle** : il s'agit de mixer plusieurs de ces techniques avec l'utilisation d'un mot de passe. Les plus répandues sont l'authentification par SMS et par mail.

Certaines banques couplent une authentification de leur application Web avec l'authentification depuis leur application mobile (sécuripass).

Section 6 Mise en place d'un contrôle d'accès

6.1 Problématique

Le **contrôle d'accès** vise à garantir qu'un utilisateur particulier a les permissions suffisantes pour :

- Créer, lire, mettre à jour ou supprimer une **ressource sécurisée** ;
- Accéder à une **fonction sécurisée** (back end d'administration par exemple) ;
- Réaliser un **processus métier** protégé.

Le principe de moindre privilège stipule qu'une tâche **ne doit bénéficier que des privilèges strictement nécessaires à l'exécution du code menant à bien ses fonctionnalités** .

Il faut donc identifier pendant la phase de spécification les privilèges juste suffisants pour chaque fonctionnalité offerte par l'application.

Le contrôle d'accès est à réaliser côté serveur.

Le contrôle d'accès doit se faire aussi au niveau du SGBD : créer des types d'utilisateur qui vont restreindre et contrôler les modifications BD.

Il est intéressant de tracer dans un log les tentatives d'authentifications et accès.

Dans le cas d'un site Web monolithique l'authentification est mémorisée dans une Session http. La session expire sur déconnexion volontaire ou suite à un hors temps.

Des conseils ici : https://www.vaadata.com/blog/fr/comment-securiser-les-systemes-dauthentification-de-gestion-de-sessions-et-de-controle-dacces-de-vos-applications-web/

Dans le cas d'une API Web, l'authentification et le contrôle d'accès classique est à remplacer par un mécanisme de jetons : les tokens.

Oauth2 est un protocole qui implémente la techno JWT

Le protocole prévoit que le client obtienne un **jeton** avec **une portée et une date de validité**. Ces jetons sont attribués par le serveur d'autorisation avec l'approbation du propriétaire de la ressource.

Bon article ici :

https://www.codeheroes.fr/2018/03/23/securiser-une-api-rest/

6.2 Un contrôle d'accès dans l'application

L'application fournie a un contrôle d'accès très réduit :

- un utilisateur non connecté peut voir les personnes, les films et s'inscrire
- un utilisateur connecté peut supprimer des personnes et des films

Il nous est demandé d'avoir 3 niveaux :

- utilisateur non connecté
- utilisateur connecté sans privilège
- utilisateur administrateur

Avec les fonctionnalités suivantes :

Le nom de l'utilisateur connecté est visible dans la barre du haut.

L'utilisateur non connecté n'a accès qu'à Home, S'inscrire, Se connecter.

L'utilisateur connecté peut en plus accéder à Personnes et Films, et à leur vue de détail

L'administrateur peut en plus supprimer une Personne et un Film, et Modifier une Personne et un film.

Il faut aussi protéger la BD.

Créer 3 profils en BD. Allouer les bons droits par profil.

Utiliser le bon profil en fonction de l'utilisateur connecté.

Tout nouvel utilisateur qui utilise la fonction s'inscrire a par défaut les droits minimaux, sans privilège.

S'il reste du temps en fin de formation ajouter une page réservée aux administrateurs. Elle permet de modifier le niveau d'accès d'une personne.

6.3 Solution

6.3.1 Base de données

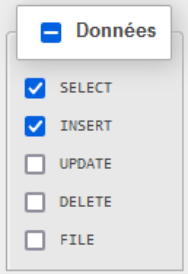
Ajouter trois types d'utilisateur :

PFNonConnecte (PF pour person film)

PFConnecte

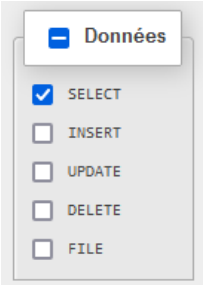
PFAdmin

PFNonConnecte a des droits réduits à l'écriture et lecture de la table users



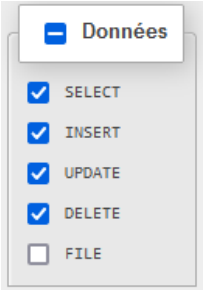

```
CREATE USER 'PFNonConnecte'@'%' IDENTIFIED WITH
mysql_native_password AS '***';GRANT SELECT, INSERT ON *.* TO
'PFNonConnecte'@'%' REQUIRE NONE WITH MAX_QUERIES_PER_HOUR 0
MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0
MAX_USER_CONNECTIONS 0;
```

PFConnecte a les droits de lecture sur users, person, film, regarde




```
CREATE USER 'PFConnecte'@'%' IDENTIFIED WITH
mysql_native_password AS '***';GRANT SELECT ON *.* TO
'PFConnecte'@'%' REQUIRE NONE WITH MAX_QUERIES_PER_HOUR 0
MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0
MAX_USER_CONNECTIONS 0;
```

PFAdmin a les droits de lecture, écriture, delete sur users, person, film, regarde




```
CREATE USER 'PFAdmin'@'%' IDENTIFIED WITH
mysql_native_password AS '***';GRANT SELECT, INSERT, UPDATE,
DELETE ON *.* TO 'PFAdmin'@'%' REQUIRE NONE WITH
MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0
MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;
```

Ajouter dans la table users une colonne niveau

```
ALTER TABLE `users` ADD `niveau` INT NOT NULL DEFAULT '1'
AFTER `password`;
```

6.3.2 Code php

Il faut que la connexion à la BD puisse se faire avec le bon niveau.

Dans la classe Database modifier connect() pour obtenir connect(int $niveau).

Section 7 Sécuriser les sessions

7.1 Problématique

Le protocole HTTP est Stateless – sans état.

Le mécanisme de session est à employer pour avoir un souvenir des données entre pages d'un même site Web.

7.2 Les bonnes pratiques en PHP

- Mettre le code session_start() comme première instruction php d'une page
- Ne jamais rendre visible (dans URL par exemple) un identifiant de session
- utiliser des cookies « HTTP Only » pour qu'ils ne soient pas manipulés par du code Javascript côté client.
  https://www.php.net/manual/en/session.configuration.php#ini.session.cookie-httponly
- Maîtriser et limiter le temps de la session. 15 Minutes est un temps moyen.
  `cookie_lifetime` est à utiliser en PHP
  exemple sur https://www.php.net/manual/en/function.session-start.php
- prévoir un moyen explicite pour fermer la session.

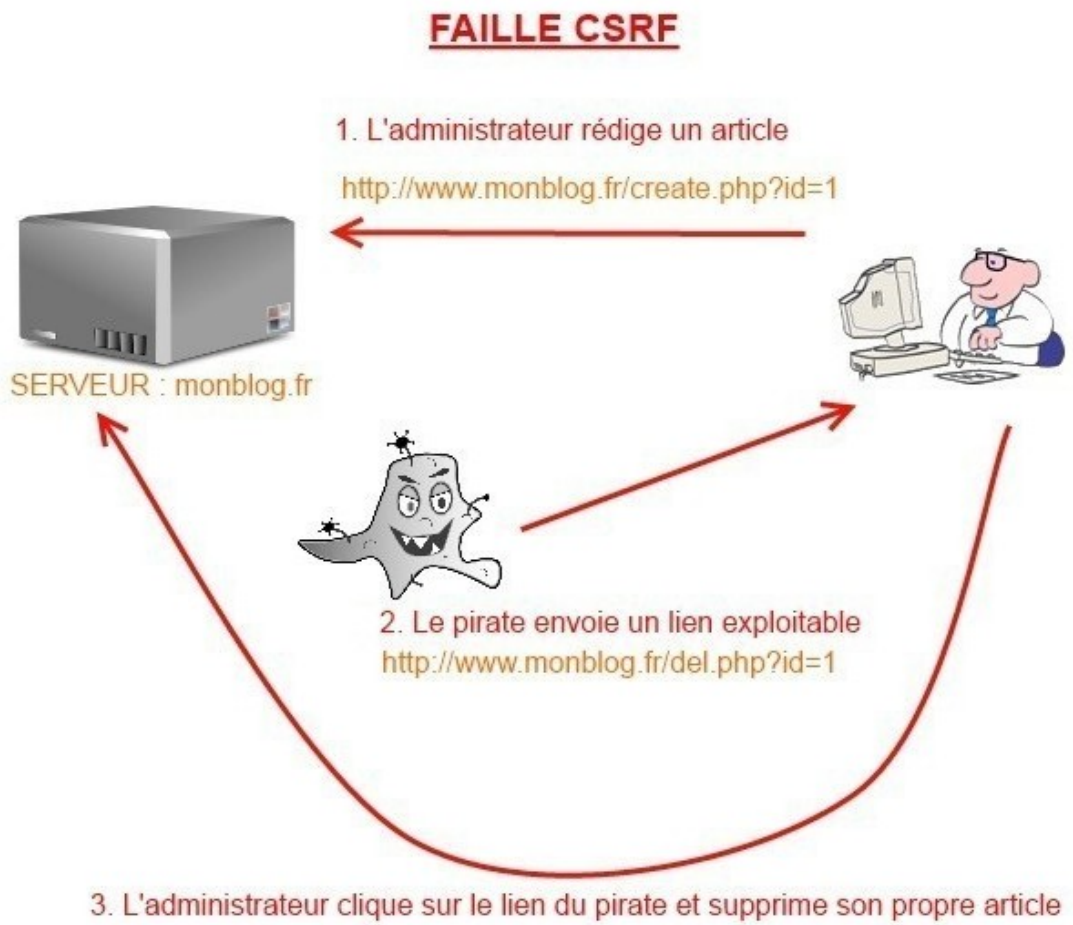
Dans l'application :

- placer HTTP Only
- limiter le temps de session à 10 mn
- permettre de fermer explicitement la session

7.3 La faille CSRF

CSRF : Cross-Site Request Forgery

Lien : https://openclassrooms.com/fr/courses/1761931-securisez-vos-applications/5702666-gerez-les-sessions-de-vos-applications-de-maniere-securisee#/id/r-5828527



La faille CSRF en 3 étapes

Vérifions que cette faille existe dans l'application.

Connectons nous à l'application. Afficher la page des Personnes.

Regardons le code source ou le bas du navigateur, copions le lien pour supprimer une personne.

Vous écrire un mail dans lequel vous copiez ce lien.

Commenter le lien par http://

Vous envoyer ce mail.

Dans le mail reçu, clic sur le lien : => la personne disparaît de l'application !

Nous allons implémenter du code pour éviter cette faille.

La solution consiste à créer un jeton au moment de la connexion et à le fournir pour les transactions qui conduisent à une modification des données.

Le code est inspiré de

https://code-boxx.com/simple-csrf-token-php/

Dans login.php ajouter dans la session une information 'token' qui contient un nombre aléatoire fourni par : `md5(bin2hex(openssl_random_pseudo_bytes(6)))`

Dans personneView.php fournir dans l'URL du bouton Supprimer ce token

Dans supprimerPersonne.php ajouter du code pour vérifier que le token existe et que celui mémorisé dans la session est le même que celui fourni dans l'URL.

Essayer le code.

Vérifier qu'un lien depuis un email ne fonctionne plus. Le hacker est sensé ne pas fournir le même token.

Section 8 Manipuler des données non fiables

8.1 Problématique

Il est capital pour vos applications de **traiter les données échangées ou stockées avec une grande prudence**, car elles portent en elles de nombreux dangers. Les sous-estimer serait une erreur colossale. Nous allons étudier dans cette partie :

1. Les mécanismes de manipulation de **données non fiables** ;
2. Les enjeux de la **protection des données** ;
3. Le traitement sécurisé des **bases de données** ;
4. La gestion des **erreurs** et des **journaux d'historisation**.

8.1.1 L'injection SQL

Il ne faut pas se fier aux données reçues en retour d'un formulaire. Un hacker peut y placer du code SQL malveillant.

Essai dans l'application :

Il reste du code à implémenter pour la modification d'une personne.

Implémenter ce code dans modifyPerson de la classe ManagePersons.

Ne pas mettre de requête préparée dans un 1er temps.

Se connecter en tant qu'administrateur.

Aller dans la vue de détail des personnes.

Vérifier que la modification d'une personne fonctionne bien.

Depuis phpMyAdmin exporter la table users car nous allons la détruire ...

Retourner dans le formulaire de modification et imaginer une saisie qui fasse un DROP de la table . (';)

Il faut donc protéger les requêtes SQL par des requêtes préparées. Dès qu'une requête récupère une information externe, la requête doit être préparée.

Modifier le code pour avoir des requêtes préparées là où cela est nécessaire.

Tester

8.2 La faille XSS

Cette fois l'intrusion se fait sous la forme de code HTML pouvant contenir du JavaScript.

Dans l'application, dans la vue de détail, saisir dans le prénom par exemple

essai

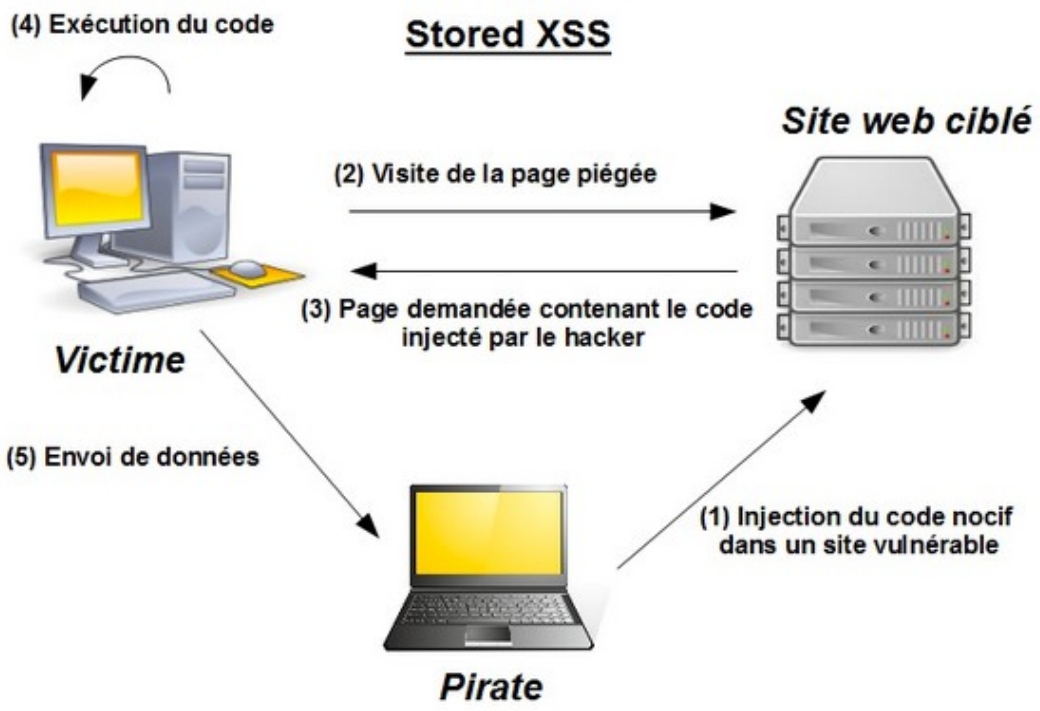
Si le texte saisi apparaît en gras dans la vue des personnes, cela indique que l'application n'est pas protégée.

Essayer alors la saisie de

<script>alert(\'vous devez une rançon à l'adresse xxx\')</script>

Tester le résultat.

Nous venons de faire une faille XSS permanente car l'injection est mémorisée en BD



Faille XSS permanente

Utiliser la fonction `htmlspecialchars()` pour neutraliser les tags html.

La fonction PHP filter_var() permet de contrôler les valeurs

https://www.php.net/manual/fr/function.filter-var.php

Il faut aussi contrôler les valeurs attendues en général et contrôler les valeurs attendues des paramètres des fonctions et méthodes.

Modifier l'application pour éviter la faille XSS.

Section 9 La protection des données

9.1 Problématique

La protection des données est une des **priorités d'un développement sécurisé**, en particulier d'un point de vue de l'utilisation des API.

Par ailleurs, un certain nombre de lois et de réglementations, comme la nouvelle réglementation européenne **RGPD,** sont mises en œuvre pour **protéger les données à caractère personnel.**

Le chiffrement de bout en bout constitue une garantie pour vos utilisateurs. Cette technique contribue aussi au développement de **la confiance entre les utilisateurs et les développeurs**.

Les échanges doivent être de bout en bout sécurisés. Le protocole TLS est à utiliser.

Les protocoles SSL/SSH protègent les données qui circulent entre le serveur et le client : SSL/SSH ne protège pas les données une fois dans la base. SSL est un protocole en ligne.

Les données sensibles doivent être stockées de façon chiffrées.

Exemple sur https://www.w3docs.com/snippets/php/how-to-encrypt-and-decrypt-a-string-in-php.html

Le RGPD

Video sur https://www.youtube.com/watch?v=iMD3pWAFXEY

Numéro de téléphone, adresse, orientation politique, préférences alimentaires... Aujourd'hui, ces données personnelles constituent une part majeure de l'économie numérique et leur utilisation est encore difficilement contrôlée.

Un nouvel outil de surveillance est mis en place : le Règlement général pour la protection des données (RGPD) .

9.2 Application

Dans notre application identifier les données personnelles et faire en sorte qu'elles ne soient plus visibles directement dans la BD.

Avant la page de connexion, afficher une page qui informe de la sauvegarde des données et demande d'accepter. Sinon la page d'accueil est affichée et on ne peut rien faire d'autre.

Section 10 Ecrire un journal pour l'application

10.1 Problématique

(extrait openclassroom)

Un ou plusieurs fichiers de log au format prédéfini sont générés en cours d'exécution et conservent des messages informant au minimum sur :

- **L'horodatage** de l'événement ;
- L'estimation de la **sévérité** de l'événement ;
- **L'identité** du compte concerné ;
- **L'adresse IP** associée avec la requête ;
- Le **résultat** (échec ou réussite) ;
- Une courte **description.**

Cette fonctionnalité est intéressante à plus d'un titre :

- L'administrateur dispose de **statistiques** sur l'utilisation d'un système ;
- Le développeur peut déceler des **défaillances** et corriger les bugs ;
- L'utilisateur peut utiliser un **journal** afin de revenir sur une panne et refaire les opérations perdues.

Dans le cadre d'une approche de **développement sécurisée**, je vous recommande de garder traces des événements liés :

- Aux **erreurs d'authentification** ;
- Aux **accès** — erreurs de contrôles d'accès, tentatives de connexion avec des données de session expirées ou invalides, échecs de connexion TLS à l'administration du site.

10.2 Implémentation dans l'application

L'instruction php suivante permet d'écrire dans un fichier de type log

```
error_log("Error message\n", 3, "/mypath/php.log");
```

Imaginer et coder une classe qui permet d'écrire des messages dans un fichier log.

On imagine 3 niveaux de log : information, warning, error . Ces niveaux sont visibles dans le message écrit dans le fichier.

Exemple de message :

2022/12/28 14:15:23 Error : this user xxxx is unknown

Utiliser ensuite cette classe pour que l'application génère des messages. Privilégier les logins et surtout ceux qui échouent.

Section 11 Ne pas oublier

11.1 Fichier .htaccess

Pour l'instant les accès aux répertoires ne sont pas protégés.

Ainsi si vous utilisez l'URL localhost/view , le navigateur va montrer l'ensemble des fichiers présents .

Il faut protéger cet accès.

Lorsque cela est possible, avec Apache, privilégier la modification du fichier httpd.conf, à la rubrique

<Directory />

Moins performant, il est possible de placer un fichier .htaccess dans les répertoires à protéger du site.

Il contient :

order deny,allow

deny from all

AuthName "Page d'administration protégée"

AuthType Basic

AuthUserFile "C:\<chemin absolu>\XAMP\htdocs\admin\.htpasswd"

Require valid-user

Créer le fichier .htpasswd

Chaque ligne contient les ref utilisateur sous la forme

nom:<mot de passe>

Dans l'env XAMPP <mot de passe> est en clair, sur site il doit être crypté. Dans ce cas la ligne

<?php echo crypt('mot de passe'); ?>

fournit la chaîne cryptée

11.2 Trace des erreurs php

Lorsque le site va être mis en production, masquer les erreurs PHP

Dans php.ini modifier pour ne pas avoir les valeurs mises en développement `error_reporting=E_ALL et display_errors=on`