UNIVERSITE VIRTUELLE DU SENEGAL
foo nekk foofu la

Sécurisation et optimisation des applications mobiles et web

M. Birahim BABOU

Séquence 3 : Chapitre 3 : Bonnes pratiques

Table des matières

Table des figures

Table des tableaux

1. Chapitre 3 : Bonnes pratiques

1.1. Les 6 aspects de la sécurité d'une application

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**.

1.1.1. L'authentification

L'authentification consiste à savoir lier, grâce à une caractéristique discriminante (un mot de passe par exemple) une identité à une entité donnée d'un système (la partie administration d'un site, un processus en particulier ou 1'ordinateur par exemple). Elle s'applique à l'utilisateur, à l'émetteur d'un message ou à l'auteur d'un document.

Pour implémenter ces fonctions, plusieurs approches sont possibles :

- authentification par **identifiant et mot de passe** ;
- authentification par **certificat ;**
- authentification par **carte** ;
- authentification **multimodale** qui associent plusieurs des méthodes précédentes.

L'authentification est probablement la notion la plus simple à comprendre, mais en tant que développeur, c'est une de celles qui vous demanderont **le plus de travail d'implémentation**.

Les problèmes potentiels à gérer sont nombreux :

- S'assurer que notre utilisateur est bien la personne qu'il prétend être ;
- Lutter contre l'espionnage, la capture et la réutilisation des mots de passe dans une autre partie du système ou sur autre application par exemple ;
- Altération des messages entre les différents acteurs du système ;
- Transférabilité du mot de passe ;

1.1.2. Le contrôle d'accès

Une fois authentifié, l'utilisateur souhaite **accéder à des fonctionnalités** offertes par l'application. Au préalable, il faut contrôler s'il a le **droit d'y accéder**.

Assurer cette fonction c'est avoir la capacité de lier une **ressource** (une base de données par exemple) avec des **droits d'accès** à cette ressource et une **entité**.

Pour gérer cette problématique, on attribue souvent à l'utilisateur un **rôle ou profil** (utilisateur, éditeur, administrateur par exemple) qui va lui octroyer des **privilèges ou des attributions** sur des **ressources** manipulées par l'application.

1.1.3. L'intégrité des données

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).

Les moyens technologiques principaux pour y parvenir sont le **calcul d'une signature** unique et caractéristique d'une ressource. Les fonctions de hachage ou le calcul de sommes de contrôle peuvent être utiles dans ce contexte.

1.1.4. La confidentialité des données

La confidentialité des données doit être assurée lors d'échange de **données sensibles** (mot de passe, données bancaires ou médicales, armées, etc.) Il s'agit de garantir que des données acquises illégalement soient inutilisables.

Au-delà des mesures organisationnelles que l'on peut mettre en œuvre (marquage, gestion particulière), 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.

1.1.5. 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, que nous verrons plus en détail plus loin dans le cours.

Cette mesure est particulièrement **complexe à mettre en œuvre**, car il est difficile de remettre en cause la bonne foi d'une personne prétextant qu'elle s'est fait dérober son identité.

1.1.6. La protection contre l'analyse du trafic

La sécurité des communications repose sur des mécanismes déjà abordés et que nous approfondirons ensuite : mécanisme **d'authentification,** de **chiffrement** et de **hachage.**

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

1.2. Méthodologie sécurisée de conception logicielle

1.2.1. Élaboration du cahier de charge

1.2.1.1. Historique de modification du document

Le cahier des charges est un document permettant de recueillir l'ensemble des processus, procédures et informations importantes pour la réalisation d'un nouveau projet d'application ou la mise à jour d'une application existante.

La première étape de ce document est l'identification de la version du document pour permettre à l'équipe projet de pouvoir suivre l'historique de modification du document.

Auteurs	Actions	Date	Commentaires

Tableau 1: Historique de modification d'un cahier de charge

1.2.1.2. Contexte

Le contexte de la mise en place de cette application, à remplir par le demandeur ou client.

1.2.1.3. Enjeux, objectifs et contraintes du projet

A remplir par le demandeur

1.2.1.4. *État des lieux/ Description de l'existant*

A remplir par le demandeur.
Revenir sur l'existant, c'est à dire :

- Le lieu d'installation de l'application (serveur physique ou en ligne)
- Existence d'une application ou pas

1.2.1.5. *Description du projet*

A remplir par le demandeur.
Revenir sur :

- le type d'application à mettre en place (web ou mobile)
-

1.2.1.6. *Attentes globales ou fonctionnalités souhaitées*

A remplir par le demandeur

1.2.1.7. *Type d'accès*

A remplir par le demandeur

1.2.1.8. *Niveau de criticité des données*

A remplir par le demandeur

1.2.1.9. *Profils des utilisateurs*

A remplir par le demandeur.

Revenir sur l'ensemble des acteurs de l'outils et les profils de chaque acteur.

Ce serait intéressant de créer une matrice de correspondance entre les acteurs et les différentes fonctionnalités. Ci-dessous un exemple de cette matrice.

ACTEURS	GESTION DES INSCRIPTIONS ET REINSCRIPTIONS					
	INSCRIPTIONS			ETUDIANTS		
Intitulé	LIST_INSCRIT	INSCRIRE_ETU	VIEW_STAT	LIST_ETU	ADD_ETU	MAJ_ETU
Administrateur	*	*	*	*	*	*
Responsable des inscriptions (scolarité)	*	*	*	*	*	*
Responsable Administratif (ENO)	*	*	*	*		
Responsable Administratif (Pôle)	*		*	*		
Agent Comptable	*		*	*		

Tableau 2: Matrice de correspondance entre les acteurs et les fonctionnalités

1.2.2. Modèle MVC

1.2.2.1. Comment fonctionne une architecture MVC ?

Il y a des problèmes en programmation qui reviennent tellement souvent qu'on a créé toute une série de bonnes pratiques que l'on a réunies sous le nom de *design patterns*.

Un des plus célèbres *design patterns* s'appelle MVC, qui signifie **Modèle - Vue - Contrôleur**.

Le pattern MVC permet de bien organiser son code source. Il va vous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

- **Modèle** : cette partie gère les *données* de votre site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc entre autres les requêtes SQL.
- **Vue** : cette partie se concentre sur l'*affichage*. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.
- **Contrôleur** : cette partie gère la logique du code qui prend des *décisions*. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).

La figure suivante schématise le rôle de chacun de ces éléments.

Tableau 3: Architecture MVC

Il est important de bien comprendre comment ces éléments s'agencent et communiquent entre eux. Regardez bien la figure suivante.

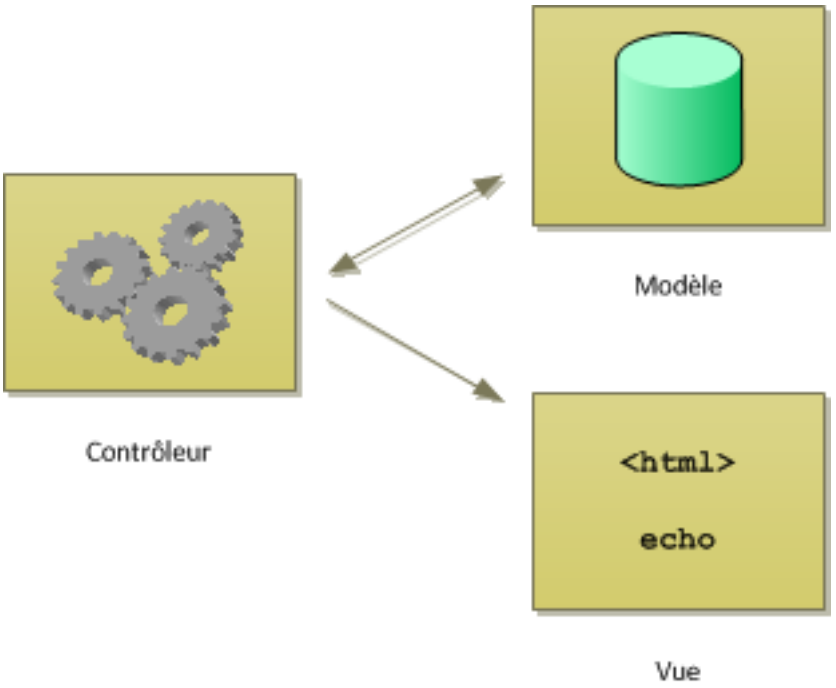

Tableau 4: Échange d'informations entre les éléments

Il faut tout d'abord retenir que le contrôleur est le chef d'orchestre : c'est lui qui reçoit la requête du visiteur et qui contacte d'autres fichiers (le modèle et la vue) pour échanger des informations avec eux.

Le fichier du contrôleur demande les données au modèle sans se soucier de la façon dont celui-ci va les récupérer. Par exemple : « Donne-moi la liste des 30 derniers messages du forum numéro 5 ». Le modèle traduit cette demande en une requête SQL, récupère les informations et les renvoie au contrôleur.

Une fois les données récupérées, le contrôleur les transmet à la vue qui se chargera d'afficher la liste des messages.

Dans les cas les plus simples, ce sera probablement le cas. Mais le rôle du contrôleur ne se limite pas à cela : s'il y a des calculs ou des vérifications d'autorisation à faire, c'est lui qui s'en chargera.

Concrètement, le visiteur demandera la page au contrôleur et c'est la vue qui lui sera retournée, comme schématisé sur la figure suivante. Bien entendu, tout cela est transparent pour lui, il ne voit pas tout ce qui se passe sur le serveur. C'est un schéma plus complexe que ce à quoi vous avez été habitués, bien évidemment : c'est pourtant sur ce type d'architecture que repose un grand nombre de sites professionnels !

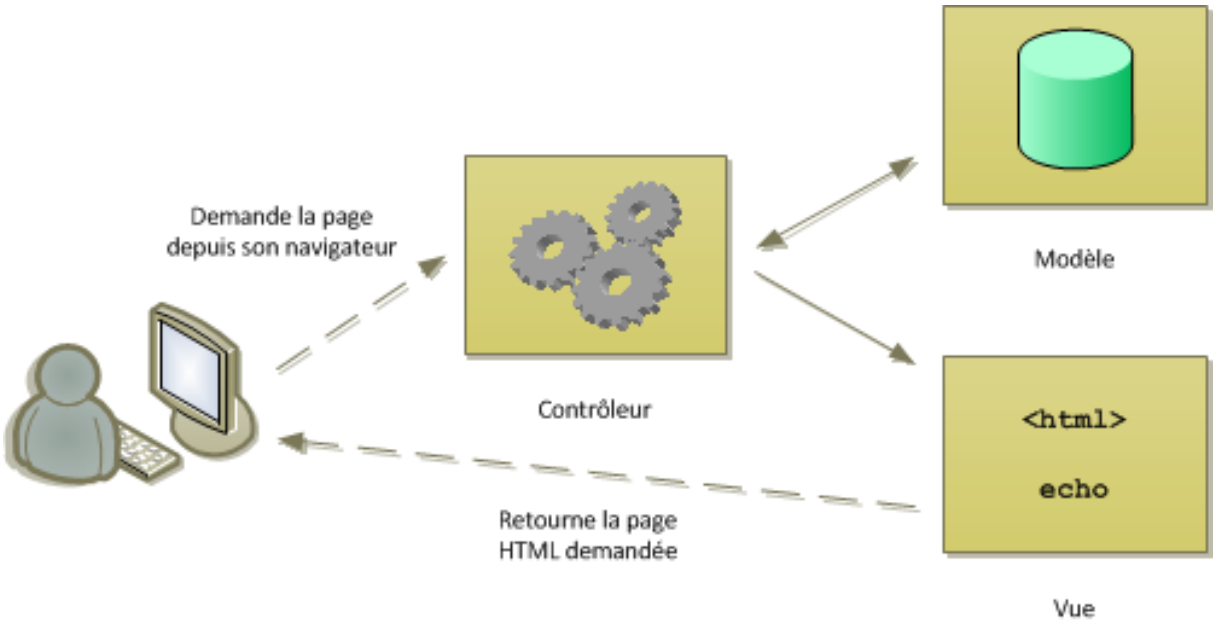

Tableau 5: La requête du client arrive au contrôleur et celui-ci lui retourne la vue

Comme exemple, on peut avoir l'architecture suivante :

- index.php : le contrôleur (chef d'orchestre)
- vues/indexView.php : la vue (page HTML...)
- modeles/model.php : le modèle (requêtes SQL...)

Source : https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php/4678736-comment-fonctionne-une-architecture-mvc

1.2.3. Cycle de vie d'un logiciel

De manière globale, on peut prendre en compte les 4 dimensions de **gouvernance,** de **conception,** de **vérification** et **d'opération.**

Chacune de ces fonctions représente une catégorie d'activités basiques du développement logiciel que les entreprises ont mises ou devraient mettre en œuvre :

1. La **gouvernance** se concentre sur les processus et les activités liés à la **gestion globale de la sécurité** par l'entreprise dans le domaine du développement informatique. Elle inclut également les impacts potentiels sur les processus métier ;
2. La phase de **construction** s'intéresse aux processus et aux activités liées à la façon dont une organisation **définit des objectifs** et **crée des logiciels** dans ses projets de développement. En général, vous y trouverez le management des produits, la collecte des exigences, les spécifications d'architecture de haut niveau, la conception détaillée et la mise en œuvre.
3. La phase de **vérification** s'attache à décrire les activités de **vérification et de tests** mises en place dans l'entreprise. On y trouve notamment la revue de conception et d'implémentation et les tests de sécurité.
4. Enfin, la phase **opérationnelle** comprend les activités liées à la gestion des déploiements des applications de l'entreprise. On y inclut souvent la **gestion des remontées d'alerte**, le contrôle sur l'environnement dans lequel évoluent les applications et les opérations standard de sécurité dans l'environnement de déploiement.

Ces 4 fonctions décrivent l'ensemble des phases dans lesquelles peut se trouver une application. À chacune de ces 4 fonctions clés, **les méthodologies de la littérature** associent :

- Des **bonnes pratiques de sécurité** ;
- Des **niveaux de maturité** pour chacune d'entre elles, afin de pouvoir en mesurer l'impact.

Pour la réglementation, la mise en place du **RGPD** en Europe et la CDP au Sénégal par exemple, bouleversent le paysage législatif pour le traitement des données à caractère personnel, et a donc un impact direct pour les développeurs.

1.2.4. Application de la sécurité à ces différentes phases

Intégrer la sécurité dans la phase de développement est indispensable pour éviter les vulnérabilités béantes dans vos applications, mais il est aussi très important d'adopter, dans **le cycle de vie complet du logiciel**, une approche où la sécurité est prise en compte à chaque étape.

Adopter une méthodologie globale aide les entreprises et les développeurs à **amoindrir les failles** dans la conception des applications. Cette méthodologie abaisse énormément le coût humain et financier d'intégration de la sécurité sur le long terme, et de correction des bugs.

Cette méthode revêtent plusieurs avantages parmi lesquels :

- La **montée en compétence de l'ensemble des équipes projets** dans le domaine de la **sécurité ;**
- Prendre en compte la sécurité dès la phase d'étude d'opportunité garantit un retour sur investissement sur le long terme ;
- **L'évaluation des pratiques existantes** de développement logiciel d'une organisation en termes de sécurité ;
- La construction d'un **plan de sécurité global** et équilibré dont bénéficieront vos applications, et adapté aux risques spécifiques auxquels l'entreprise fait face ;
- **L'amélioration concrète** d'un plan global existant ;
- **La formalisation**, la **mesure** et le **suivi** des activités liées à la sécurité dans toute l'entreprise.

1.3. Cryptage SSL

La première preuve de sécurité informatique que vos visiteurs peuvent voir sur votre site internet est la **présence du sigle HTTPS** accolé à votre URL. Cela leur garantit que les communications avec votre plateforme sont cryptées grâce à la **présence d'un certificat SSL**. Son fonctionnement est simple : il met en place des clés de chiffrement privées pour les échanges entre l'ordinateur et votre serveur, les rendant illisibles pour tout observateur extérieur. Personne ne pourra ainsi copier les données bancaires de vos clients lorsqu'ils achètent un produit.

Cet outil de sécurité de site web est aujourd'hui très largement répandu, et constitue la **base de la sécurisation d'une plateforme**.

1.3.1. HTTPS

1.3.2. FTPS

1.3.3. SSH

1.4. Sauvegardes et mises à jour fréquentes

Pour améliorer encore la sécurité des données, vous pouvez réaliser deux simples actions : sauvegarder votre site application et mettre à jour vos logiciels.

De nouvelles failles de sécurité sont découvertes presque tous les jours par les éditeurs de logiciel. Afin de contrecarrer les potentielles utilisations malveillantes de toute vulnérabilité, ils proposent régulièrement des mises à jour de leurs produits à tous leurs utilisateurs.

Il est essentiel de les installer sur votre application dès qu'elles apparaissent, quel que soit le logiciel, pour limiter le risque d'intrusion dans les données de vos clients. Il peut s'agir d'un élément aussi essentiel que le CMS ou tout simplement du thème utilisé pour habiller votre site internet.

En matière de sécurité informatique, la sauvegarde de vos données n'est pas une solution directe, mais il s'agit tout de même d'une étape fondamentale. En téléchargeant régulièrement votre base de données sur un support de stockage extérieur, disque dur externe ou celui de votre ordinateur, vous vous assurez de pouvoir remettre en ligne une version saine de votre site internet si des pirates exploitaient une faille de sécurité. Nous vous conseillons de faire plusieurs copies de cette sauvegarde sur des supports différents, en cas d'infection de l'un d'eux.

1.5. Éviter d'exposer les interfaces d'administration et autres applications de gestion à distance

Les mécanismes d'administration des applications web et mobiles sont par nature des éléments sensibles dont l'exposition doit être limitée.

Ci-dessous quelques règles à adopter pour une bonne sécurisation de vos applications :

- Pour administrer les applications web à distance, Il est recommandé d'utiliser les protocoles sécurisés tels que SSH, SFTP et HTTPS.
- Il est préférable de restreindre l'accès à l'administration des sites web aux seuls postes d'administration autorisés. Il est souhaitable de doter le serveur web d'une autre interface réseau réservée pour les connexions d'administration.
- Éviter d'exposer les interfaces d'administration (par exemple : phpmyadmin, webmin, cpanel, etc.) à l'extérieur. Au cas où cette solution est nécessaire, il est important de :
    - Utiliser un VPN avec chiffrement ;
    - Utiliser des mots de passe fort ;
    - Activer un système de connexion à la demande ;
    - Journaliser l'activité des accès distants ;
    - Contrôler des tentatives d'accès.

1.6. La réglementation

1.6.1. La protection des données personnelles

2. Projet

- Élaboration d'un cahier des charges pour la mise en place d'une application web ou mobile sécurisée
- Mise en place de l'application validée
- Test de la vulnérabilité de cette application avec des captures d'écran illustratives
- Présentation de l'application et de la solution de sécurité

Conclusion

La sécurité joue un rôle très important dans un système d'information. La maitrise des applications et système existants permet de maitriser la sécurité de son système.

Au cas où c'est nécessaire, il faudra adopter les actions suivantes pour limiter les accès distants :

- accéder à travers un VPN avec un chiffrement
- utiliser des mots de passe forts
- Journaliser et contrôler les accès à distance

Webographie

1. https://www.alphorm.com/tutoriel/formation-en-ligne-hacking-et-securite-lessentiel

2. https://openclassrooms.com/fr/courses/1761931-securisez-vos-applications
3. https://fr.godaddy.com/blog/securite-informatique-outils-pour-votre-site/
4. https://fr.slideshare.net/hellosct1/comprendre-la-securite-web-77763122
5. https://web.developpez.com/tutoriels/web/failles-securite-application-web/
6. https://www.dgssi.gov.ma/sites/default/files/attached_files/guide_appli_web_v23-12-2015.pdf
7. https://asiq.org/wp-content/uploads/2016/02/La-s%C3%A9curit%C3%A9-applicative-OWASP-diffusion.pdf
8. https://owasp.org/www-pdf-archive/OWASP_Top_10_2007_-_French.pdf
9. https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es

https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php/4678736-comment-fonctionne-une-architecture-mvc