Bases de données

Bases de données pour le Web & Sécurité Web

Pierre Senellart

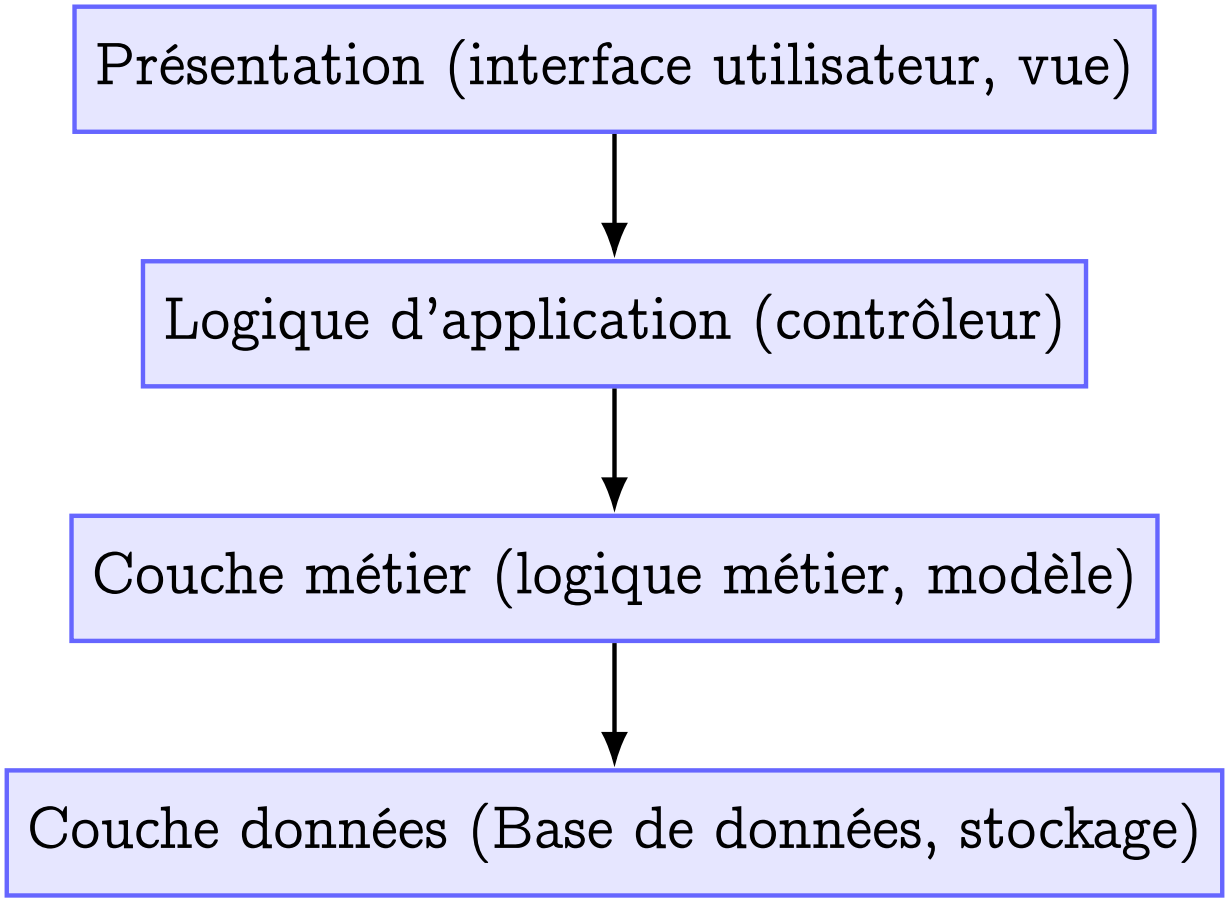
7 mai 2025

Plan

SGBD et applications Web

- La très grande majorité des sites Web repose sur du stockage des données dans un SGBD (très majoritairement relationnel)
- Réciproquement, la très grande majorité des applications développées au-dessus d'une base de données sont des applications Web (y compris pour des applications non accessibles sur le Web public, mais sur un Intranet, en local au sein d'une application ou d'un client léger...)

Architecture multi-niveaux (1/2)



Architecture multi-niveaux (2/2)

Principe de génie logiciel divisant un système logiciel en plusieurs couches (ou *tiers* en anglais) :

Une couche de présentation : chargée de fournir une interface utilisateur avec visualisation et interactions

Une couche applicative : avec les fonctionnalités permettant de contrôler l'application

Une couche métier : avec l'implémentation de la logique de l'application, du modèle des données (contraintes complexes, fonctions élémentaires sur les données...)

Une couche d'accès aux données : avec la gestion des données de l'application

Couche d'accès aux données

- Implémentée par le SGBD ! Pas forcément un SGBD relationnel, peut aussi être un SGBD RDF, clef–valeur, orientée documents... mais le rôle du SGBD est d'implémenter la couche d'accès aux données
- L'interface entre couche d'accès aux données et couche métier se fait via un langage de requêtes (SQL pour les SGBD relationnel) ou par une abstraction programmatoire de ce langage (par exemple, un ORM, voir plus loin)

Connecteurs pour SGBD

- Connecteurs (bibliothèques de fonctions) différentes pour les différents SGBD existants : MySQL, PostgreSQL, Oracle, etc.
- Certains langages de programmation (p. ex., Java) ont une interface uniforme d'accès à ces connecteurs logiciels (JDBC) intégrée au langage ; d'autres nécessitent d'utiliser des bibliothèques avec des API différente pour chaque SGBD
- Certaines bibliothèques tierces fournissent une abstraction uniforme aux différents connecteurs des SGBD
- Certains frameworks Web (voir plus loin) intègrent une bibliothèque d'accès uniforme ou un ORM

Exemple : psycopg2 en Python

Extrait de la documentation de psycopg2 :

```
import psycopg2

conn = psycopg2.connect("dbname=test user=postgres")
cur = conn.cursor()

cur.execute("CREATE TABLE test (id serial PRIMARY KEY,"
            "num integer, data varchar);")

cur.execute("INSERT INTO test (num, data) VALUES (%s, %s)",
     (100, "abc'def"))

cur.execute("SELECT * FROM test;")
print(cur.fetchone())

conn.commit()
```

ORM : Object Relational Mapping

- Couche logicielle abstrayant les accès à la base de données sous forme d'accès à des objets (de POO)
- On définit au sein des classes la manière dont chaque classe sera stockée dans la base, les types des attributs, les index, etc.
- Puis l'ORM se charge de transformer tout accès à un objet en les ordres SQL correspondants, de maintenir un cache mémoire, etc.
- Pratique, mais a un fort surcoût et on perd (sauf à faire des requêtes SQL « à la main ») la capacité de requêtes complexes, puisque les seules manipulations autorisées par l'ORM sont celles correspondant à des manipulations d'objet

Exemple : SQLAlchemy

Extrait de la documentation de SQLAlchemy :

```
class User(Base):
    __tablename__ = "user_account"

    id: Mapped[int] = mapped_column(primary_key=True)
    name: Mapped[str] = mapped_column(String(30))
    fullname: Mapped[Optional[str]]

    addresses: Mapped[List["Address"]] = relationship(
        back_populates="user", cascade="all, delete-orphan"
    )

    def __repr__(self) -> str:
        return (f"User(id={self.id!r}, name={self.name!r},"
                f"fullname={self.fullname!r})")
```

Couche métier

- Implémentée dans un langage de programmation arbitraire (ou au sein du *modèle* d'un framework Web)
- Une partie de la couche métier peut aussi avoir sa place au sein du SGBD : fonctions définies par l'utilisateurs implémentant une logique métier écrites dans des procédures stockées, et surtout contraintes sur les données et déclencheurs (voir cours sur les vues)
- L'interface entre couche métier et couche applicative se fait via des appels de fonctions

Frameworks d'applications Web

- Framework : ensemble d'un langage de programmation, d'une bibliothèque de fonctions, d'outils externes, de bonnes pratiques à suivre...
- Permet d'abstraire la création d'une page Web
- Suit en général le motif MVC (voir transparent suivant)
- Inclut parfois la génération de code JavaScript côté client pour créer directement une application Web fortement dynamique (p. ex., validation de formulaire)
- Fortement recommandé pour créer des applications complexes
- Frameworks populaires : `https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/`

Motif MVC

- Principe de génie logiciel, utilisé dans d'autres domaines
- Particulièrement adapté au cas des applications Web !
- Séparation propre entre :

  Modèle : données manipulées par l'application et fonctions de manipulation de ces données ; réutilisable pour d'autres applications Web

  Vue : présentation des données ; facilement échangeable pour changer l'apparence et la structure du site

  Contrôleur : contrôle la manière dont l'utilisateur interagit, au travers de la vue, avec les données du modèle

CMS

- CMS (Content Management System)
- Créer des sites Web sans aucun développement
- Fonctionnalités :
  - édition simplifiée de page (syntaxe wiki ou bbcode, ou contrôle JavaScript texte enrichi) ;
  - ajout de contenu externe (images, documents annexes, etc.) ;
  - gestion d'utilisateurs, contrôle d'accès, etc. ;
  - modules de gestion de forums, de blogs ;
  - thèmes graphiques prêt à l'emploi ;
  - contrôle de version.
- Suivant les CMS, extensions pouvant être nombreuses
- Certains CMS spécialisés : blogs, commerce électronique, forums, etc.
- Parts de marché : `https://w3techs.com/technologies/history_overview/content_management`

Couche applicative

- Implémentée dans un langage de programmation arbitraire, avec interfaçage avec le serveur Web ou parfois un serveur d'applications Web
- Souvent au sein du *contrôleur* d'un framework Web
- L'interfaçage entre couche applicative et couche de présentation se fait d'une part par la définition d'une correspondance entre URL et fonctions du contrôleur, d'autre part par le chargement au sein de fonctions du contrôleur des *vues* appropriées

Serveur Web

- Logiciel dont le rôle est de traiter les requêtes HTTP en renvoyant du contenu statique ou dynamique produit par une application Web
- Peuvent soit fournir de nombreuses fonctionnalités pour s'interfacer avec des langages de programmation, des frameworks Web, etc., soit au contraire être le plus léger possible pour traiter rapidement les requêtes et déléguer à un autre logiciel (un autre serveur Web ou un serveur d'applications Web) la gestion des applications Web
- Parts de marché : `https://www.netcraft.com/blog/january-2025-web-server-survey/`
- Serveur d'application Web : variante d'un serveur Web destiné au déploiement d'applications au sein d'un environnement Web uniforme (Java pour Tomcat, JavaScript pour Node.js, Lua pour OpenResty...)

Couche de présentation

- Implémentée par le navigateur Web qui effectue un rendu graphique des *vues* fournies par le contrôleur (en HTML et CSS), des interactions au sein du navigateur (principalement en JavaScript) et des interactions avec le contrôleur (via des requêtes HTTP)
- La vue peut être décrite par un système de templates de page Web, permettant de générer des pages avec des structures propres

Plan

Sécurité côté client

- N'importe qui peut créer un site Web et s'arranger pour qu'il soit référencé par les moteurs de recherche... y compris des personnes malveillantes.
- Aucune raison de faire confiance *a priori* aux créateurs d'un site que l'on visite.
- Les navigateurs sont censés fournir un environnement protégé (bac à sable) dans lequel le code HTML est rendu, les scripts JavaScript sont exécutés, etc.
- Mais cette protection n'est pas toujours parfaite.

Same-Origin Policy (1/2)

- Principe général d'interaction entre contextes d'exécution JavaScript : deux scripts (dans différentes fenêtres ou frames du navigateur) ne peuvent interagir que si l'URL du contexte correspondant a le même protocole, port, et nom de domaine
- En pratique, de nombreuses subtilités
- Possibilité de communiquer entre deux scripts qui collaborent avec l'API postMessage
- AJAX est restreint de manière similaire

Same-Origin Policy (2/2)

- Pas de restriction d'origine pour :
  - le chargement d'images, vidéos, et applets
  - le chargement de feuilles de style CSS
  - les <**iframe**> HTML
  - le chargement d'un script JavaScript avec la balise <**script**>
- Les cookies fonctionnent de manière plus libérale (la notion de domaine est étendue, pas de contrôle du port ou du protocole) pour leur écriture/lecture, mais peuvent être restreints à un chemin

Risques côté client

- Mauvais fonctionnement des logiciels
- Divulgation de données confidentielles (mots de passe, numéros de carte de crédit, adresses e-mail, etc.)
- Perte de données locales (vandalisme, rançonnement)
- Réutiliser les identifiants d'un utilisateur auprès d'un autre site pour envoyer des mails malicieux, passer des ordres de virement, faire des achats, etc.
- Mise à disposition des ressources d'un ordinateur local au sein d'un botnet :
  - Stockage de données illégales
  - Relais pour d'autres formes d'attaques
  - Envoi de spams

Failles de sécurité du navigateur

Problème

Une erreur de programmation du navigateur Web rend possible des comportements non voulus (du moins grave, un comportement un peu erratique ou un plantage, au plus grave, la prise de contrôle de l'ordinateur par un ordinateur distant).

Solution

Mettre à jour son navigateur très régulièrement (par exemple via les mises à jour automatiques du système d'exploitation ou du logiciel lui-même).

Failles de sécurité d'un autre composant

Problème

Une erreur de programmation d'une extension d'un navigateur Web (blocage de publicité, plug-in, traduction des menus, outils tiers...) rend possible des comportements non voulus.

Solution

Mettre à jour toutes les extensions concernées le plus régulièrement possible (peut devenir compliqué). Désinstaller les extensions non utilisées. S'informer des problèmes de sécurité les plus importants. Favoriser les extensions dont les mises à jour sont intégrées à celle du système d'exploitation (ou du navigateur).

Exécution de code malveillant

Problème

Un utilisateur lance une application téléchargée depuis une page Web, ceux-ci pouvant avoir n'importe quel comportement malveillant.

Solution

Faire attention aux messages d'avertissement du navigateur ! Et réfléchir.

Clickjacking

Problème

Un utilisateur visitant un site malveillant est conduit à cliquer à un endroit dangereux (avertissement du navigateur, zone d'un frame appartenant à un site de confiance, etc.) en masquant la zone où le clic va avoir lieu jusqu'au dernier moment.

Solution

Timing entre affichage et clic sur une action importante dans les navigateurs ; se méfier des comportements bizarres du pointeur de la souris dans un site

Hameçonnage (Phishing)

Problème

Via un lien (dans un e-mail, sur le Web...), un internaute est amené sur une page Web qu'il pense être celle d'un site auquel il fait confiance (banque en-ligne, commerce électronique...), et se voit demander ses identifiants de connexion. Le site n'est qu'une imitation du site officiel, et ces identifiants sont ainsi divulgués à des individus malveillants.

Solution

Toujours contrôler scrupuleusement l'URL d'un site auquel on parvient via un lien. Dans le cas où un site manipule des informations sensibles ou permet de réaliser des opérations sensibles, ne l'utiliser que sous HTTPS, en vérifiant le certificat SSL (les navigateurs récents affichent l'identité validée du propriétaire du site dans la barre d'adresse). Faire preuve de bon sens !

Capture de paquets IP

Problème

Sur un réseau local, ou sur un réseau WiFi non chiffré (ou avec un chiffrement WEP simple à casser), il est possible à un attaquant de regarder le contenu des paquets IP en clair, contenant l'ensemble de la communication entre le navigateur et le serveur Web, y compris l'ensemble des paramètres HTTP, etc. Ce problème existe aussi, mais de manière moins importante, hors du cadre d'un réseau local ou sans fil.

Solution

Ne pas utiliser HTTP pour transmettre des informations sensibles au travers d'Internet, d'autant plus dans le cadre d'un réseau local ou sans fil. HTTPS, un autre protocole permettant l'envoi chiffré de messages sur le Web (et ayant d'autres fonctionnalités avancées par rapport à HTTP), doit être utilisé.

Usurpation de session

Problème

Utilisation d'une des techniques présentées dans ce cours (en particulier, XSS ou capture de paquets IP) pour récupérer l'identifiant de session d'un utilisateur (identifiant ordinairement stocké dans un cookie), pour se faire passer pour lui.

Solution

Résoudre les autres problèmes ! Côté serveur, prévoir la possibilité de terminer la session dès que celle-ci n'est plus nécessaire.

Plan

Sécurité côté serveur

- Web : environnement hostile
- À moins de contrôler l'accès même au serveur Web, n'importe qui peut avoir accès au site Web... y compris des personnes malveillantes.
- Certaines choses sont de la responsabilité de l'administrateur (ex., avoir un serveur Web mis à jour régulièrement), le reste est de la responsabilité du webmestre.

Risques côté serveur

- Divulgation de données confidentielles (mots de passe, numéros de carte de crédit, adresses e-mail, etc.) des utilisateurs
- Mise en danger des utilisateurs
- Perte de données locales (vandalisme, rançonnement)
- Mise à disposition des ressources du serveur au sein d'un botnet :
  - Stockage de données illégales
  - Relais pour d'autres formes d'attaques
  - Envoi de spams

Validation des paramètres HTTP

Il est possible d'imposer un certain nombre de restrictions sur les données pouvant être envoyées dans un formulaire, côté client :

- maxlength sur un <**input type**="text"> impose une taille maximale aux champs de saisie texte
- Les <**select**>, bouton radio, case à cocher obligent à choisir une (ou plusieurs) des valeurs proposées
- Un champ caché <**input type**="hidden"> ou non modifiable (readonly) fixe la valeur d'un paramètre
- Du code JavaScript de validation peut faire des vérifications arbitraires

Validation des paramètres HTTP

Il est possible d'imposer un certain nombre de restrictions sur les données pouvant être envoyées dans un formulaire, côté client :

- maxlength sur un <**input type**="text"> impose une taille maximale aux champs de saisie texte
- Les <**select**>, bouton radio, case à cocher obligent à choisir une (ou plusieurs) des valeurs proposées
- Un champ caché <**input type**="hidden"> ou non modifiable (readonly) fixe la valeur d'un paramètre
- Du code JavaScript de validation peut faire des vérifications arbitraires

Mais vrai seulement si le client Web joue le jeu. Très facile à contourner (désactivation de JavaScript, modification de page).

Ne jamais faire confiance aux données envoyées par un client Web !

Toujours (re)faire les validations côté serveur.

Injection de code HTML

Problème

Un utilisateur peut entrer, à l'intérieur d'un paramètre HTTP destiné à être affiché, du code HTML (et donc également des indications de style CSS, du code JavaScript...). Il modifie ainsi le code de la page HTML produite. Si ce paramètre est stocké pour être réaffiché (ex. commentaires de blog), ce code influe sur l'apparence du site pour d'autres utilisateurs.

Exemple

Supposons que le paramètre HTTP `login` contienne :
`<div style='color: red'>` dans du code PHP :

```
echo "Bonjour ".$_REQUEST["login"]." !";
```

Solution

Utiliser les fonctions de protection des caractères spéciaux HTML (en PHP, htmlspecialchars).

XSS (Cross-Site Scripting)

Problème

Cas particulier de l'attaque précédente : insertion de code JavaScript dans une page HTML, qui sera réaffiché par d'autres utilisateurs ; le code JavaScript « vole » les informations saisies par l'utilisateur pour les transmettre à un autre site.

Solution

Comme avant, utiliser htmlspecialchars, en particulier quand un texte saisi par un utilisateur est destiné à être affiché.

XSRF (Cross-Site Request Forgery)

Problème

Un site tiers charge (p. ex., dans un <**iframe**>) une page du site attaqué. Le navigateur accède à cette page en étant authentifié comme l'utilisateur original, ce qui permet au site attaquant d'accomplir une action à la place de l'utilisateur (p. ex., passer une commande) ou de déduire des informations sur l'utilisateur

Solution

- Exiger une requête POST pour tout comportement ayant un effet de bord
- Empêcher l'inclusion d'une page dans une autre (avec l'en-tête HTTP `X-Frame-Options`)
- Faire en sorte que l'accès à cette page soit contrôlé par des tokens uniques non devinables

Injection de code SQL

Problème

Un utilisateur peut modifier une requête SQL en mettant des guillemets simples dans une variable à partir de laquelle sera construite la requête.

Exemple

Supposons que `$p` vale « `' OR 1=1 --` » dans :

```
$result=pg_query("SELECT * FROM T WHERE passwd='$p'");
```

Solution

Ne jamais construire de requête de cette façon et utiliser les requêtes préparées. Quand on ne peut vraiment pas faire autrement, utiliser les fonctions de protection des caractères spéciaux SQL (p. ex., `pg_escape_string` en PHP).

Injection de ligne de commande

Problème

Un utilisateur peut modifier les programmes externes appelés côté serveur (par exemple, en PHP, appelés à l'aide des fonctions `system`, `exec`, `passthru`...)

Exemple

Supposons que `$rep` contienne « `&& cat /etc/passwd` » dans le code PHP : `passthru("ls $rep");`

Solution

Éviter l'appel à des programmes externes depuis le serveur Web. Vérifier soigneusement le contenu des lignes de commande. En PHP, utiliser escapeshellcmd ou escapeshellarg pour protéger les caractères spéciaux pour le shell.

Traversée de répertoires (Directory traversal)

Problème

Un utilisateur peut, lors de l'accès à des fichiers sur le serveur (par exemple, en PHP, avec `fopen`, `readfile`...), en utilisant '/', '..', accéder à des fichiers auquel il n'est pas censé avoir accès.

Exemple

Supposons que `$fichier` contienne « `../../../../../etc/passwd` » dans le code PHP : `readfile($fichier);`

Solution

Vérifier que l'argument des fonctions accédant à des fichiers ne pointe pas vers des fichiers auxquels on ne souhaite pas donner accès (par ex., vérifier qu'il n'y a pas de '/' à l'intérieur).

Concurrence critique (Race condition)

Problème

Un attaquant peut produire un comportement inattendu côté serveur en exploitant une faille de raisonnement qui suppose qu'un bloc d'instructions sera exécuté en une seule fois, sans être en concurrence avec d'autres instructions.

Exemple

Un script PHP récupère le plus grand entier stocké dans une table SQL, l'augmente de un, sauvegarde un fichier avec pour nom cet entier, et ajoute une ligne correspondante dans une table SQL. Il y aura concurrence si deux scripts s'exécutent simultanément, et que les deux consultent la table pour connaître le plus grand entier avant d'avoir ajouté une ligne dans celle-ci.

Solution

Bien réfléchir aux cas de concurrence critique. Utiliser les transactions du SGBD, utiliser des verrous sur les fichiers.

Déni de service (DOS, Denial Of Service)

Problème

Attaquer un site Web (ou un autre service sur Internet) en exigeant du serveur plus que ce qu'il ne peut servir (très grand nombre de connexions, calculs coûteux...)

Solution

Essentiellement de la responsabilité de l'administrateur du site, mais le webmestre peut prévenir certaines attaques en 1) évitant les fichiers trop lourds à télécharger 2) évitant les calculs coûteux inutiles côté serveur.

Cassage de mots de passe par force brute

Problème

Les mots de passe peu sûrs (noms propres ou mots du dictionnaire sans ou avec petites variations, très courts) peuvent être cassés par force brute, en explorant un à un une liste de mots de passe possibles ; c'est d'autant moins coûteux si on arrive à se procurer l'ensemble des mots de passe hachés.

Solution

Imposer aux utilisateurs (et à soi-même !) des mots de passe sûrs. Ne pas divulguer un fichier de mots de passe, même chiffrés. Surveiller les accès à un site Web visant à essayer une liste de mots de passe, et les bloquer.

Ingénierie sociale (Social engineering)

Problème

Probablement la plus utilisée des attaques : exploiter une faille non pas dans un quelconque logiciel, mais dans le raisonnement humain ! Pousser un utilisateur honnête à donner des informations confidentielles, etc.

Solution

Garder en tout un esprit critique, faire preuve de bon sens, et ne pas se laisser abuser par une méconnaissance technique !

Autres règles de bons sens

Afin de prévenir des attaques, ou d'en limiter la conséquence :

- Utiliser des algorithmes de chiffrement reconnus, pas du bricolage.
- Ne jamais stocker de mots de passe dans une base de données, mais uniquement un hash cryptographique non réversible (par exemple, bcrypt).
- Ajouter du sel dans les mots de passe hachés, pour éviter les attaques par précalcul de hachés.
- Ne jamais stocker de numéros de cartes de crédit ou autres informations sensibles dans une base de données.
- Faire en sorte que le serveur Web (logiciel) ait des droits d'accès limités à l'ordinateur sur lequel il tourne.
- Ne pas faire une confiance aveugle à du code tiers.
- Réfléchir et se mettre dans la peau d'un attaquant

Plan

Références

- *The Tangled Web : A Guide to Securing Modern Web Applications*, Michael Zalewski, pour comprendre la manière dont les technologies du Web interagissent vis-à-vis de la sécurité (un peu daté)
- *The Art of Deception*, Kevin Mitnick, pour se sensibiliser aux dangers de l'ingénierie sociale
- *Hacking, the Next Generation*, Nitesh Dhanjani, Billy Rios, Brett Hardi (O'Reilly) pour des exemples concrets d'attaques modernes
- Outils développeurs des naviagateurs, pour étudier et analyser les sites côté client
- Sites de challenges de hacking comme `http://www.hackthissite.org/` pour se mettre dans la peau d'un attaquant

Extraits du code pénal

Article 323-1

Le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni de deux ans d'emprisonnement et de 60 000 euros d'amende.
Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de trois ans d'emprisonnement et de 100 000 euros d'amende.

Article 323-2

Le fait d'entraver ou de fausser le fonctionnement d'un système de traitement automatisé de données est puni de cinq ans d'emprisonnement et de 75 000 euros d'amende.

Article 323-3

Le fait d'introduire frauduleusement des données dans un système de traitement automatisé ou de supprimer ou de modifier frauduleusement les données qu'il contient est puni de cinq ans d'emprisonnement et de 150 000 euros d'amende.