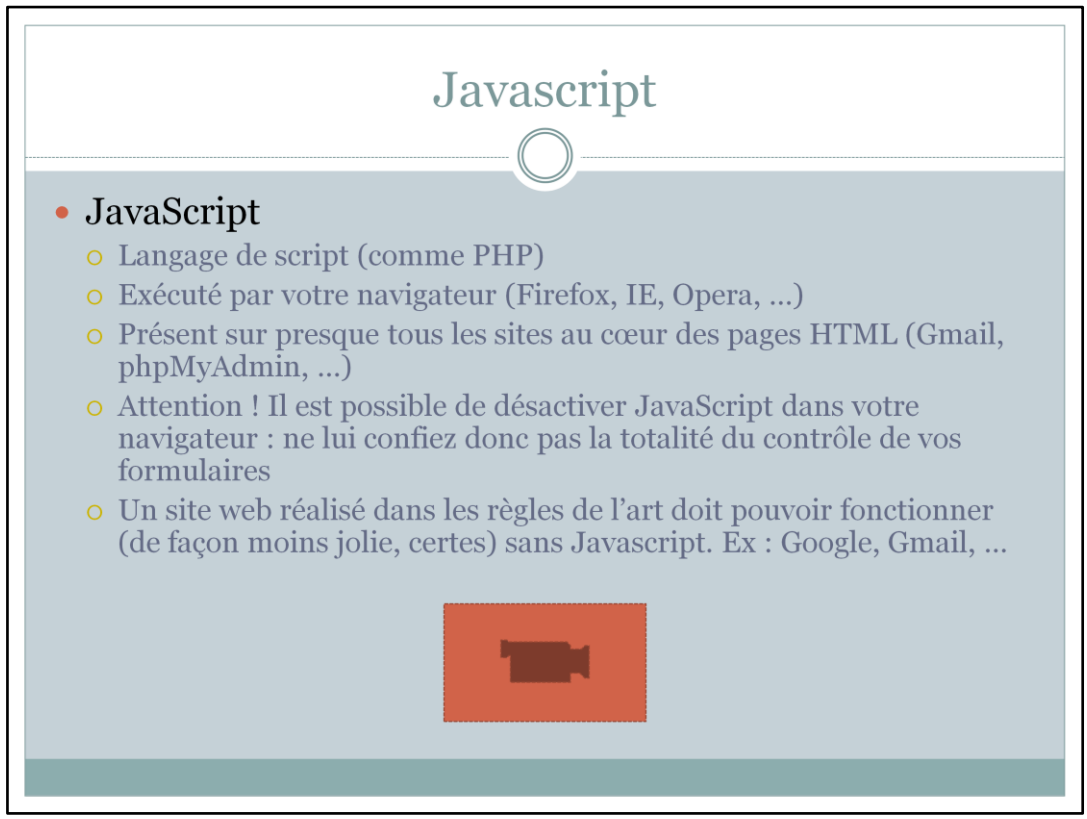
Un peu de sécurité

MODEX WEB
BAPTISTE DESPREZ



Pour désactiver Javascript sur Firefox (Windows) et voir l'impact sur les sites, allez dans :

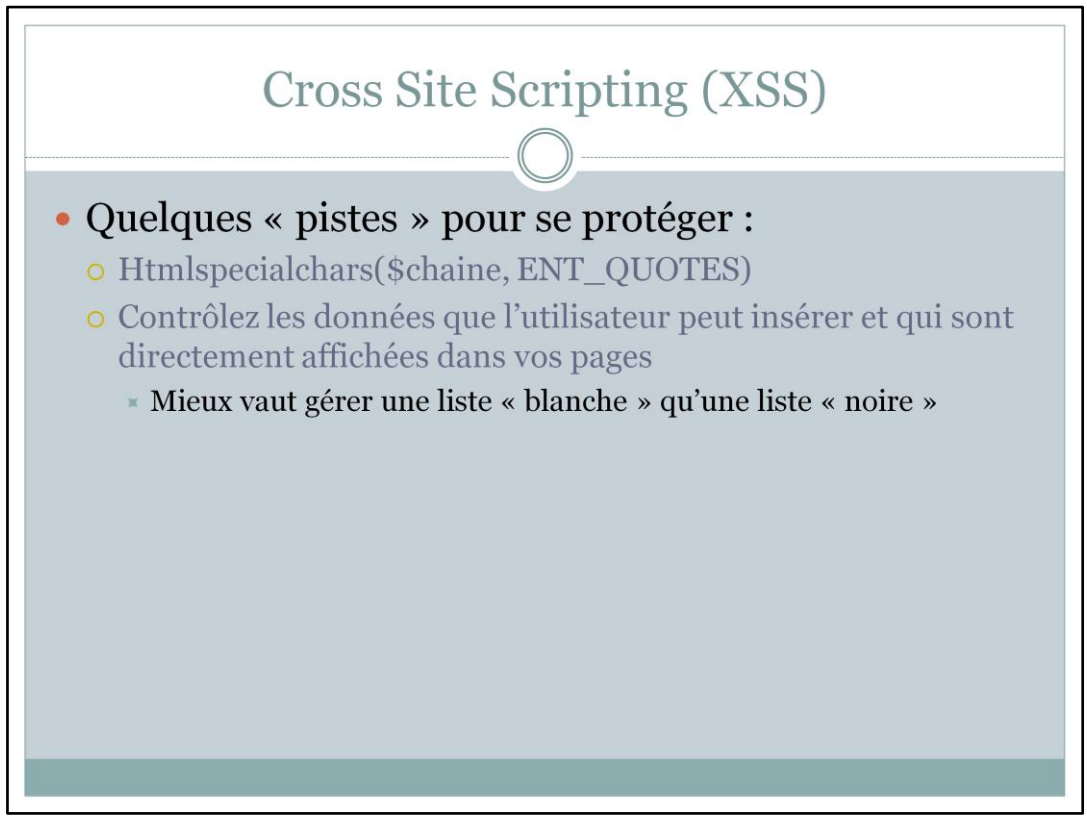
Outils -> Options… -> Contenu et décochez « Activer Javascript »

La plupart des sites modernes fonctionnent avec du Javascript. Gmail est un bon exemple.

Cross Site Scripting (XSS)

- En quoi ça consiste ?
  - Le XSS consiste à exploiter une faiblesse d'un site web qui contrôle mal les données insérées par les utilisateurs
- Les risques
  - Un utilisateur malveillant peut insérer son propre code HTML ou JavaScript
  - Très utile dans les applications qui affichent les messages des utilisateurs (forums, …) pour récupérer des informations personnelles ou rediriger vers son site pirate ou simplement faire planter le service (cf Twitter)

http://localhost/xss/search.php?q="<script>alert(document.cookie)</script>&s=0&e=10
http://localhost/xss/search.php?q="<script>window.location.replace("http://www.lix.polytechnique.fr/~rossin/")</script>&s=0&e=10



• La fonction php htmlspecialchars (http://fr2.php.net/manual/fr/function.htmlspecialchars.php) vous permet de convertir en HTML les caractères réservés à PHP.

Les remplacements effectués sont :

"*&*" (et commercial) devient "*&*"
"*"*" (guillemets doubles) devient "*"*" lorsque **ENT_NOQUOTES** n'est pas utilisée.
"*'*" (guillemet simple) devient "*'*" uniquement lorsque **ENT_QUOTES** est utilisée.
"<" (inférieur à) devient "*<*"
">" (supérieur à) devient "*>*"

Elle est très utile dans le cas de formulaires, et vous aidera à vous protéger de la plupart des XSS.

• D'une manière plus générale, il faut TOUJOURS contrôler toutes les données que les utilisateurs peuvent insérer.

Cela concerne aussi bien les formulaires que les paramètres que vous passez par $_GET pour l'affichage de votre page.

Dans le cas de la page http://mapage/index.php?id=5, n'importe quel utilisateur peut modifier le 5 et mettre ce qu'il veut !

Si vous envoyez par GET ou POST ou autre un entier, vérifiez que vous recevez un entier. C'est simple et ça ne coûte rien.

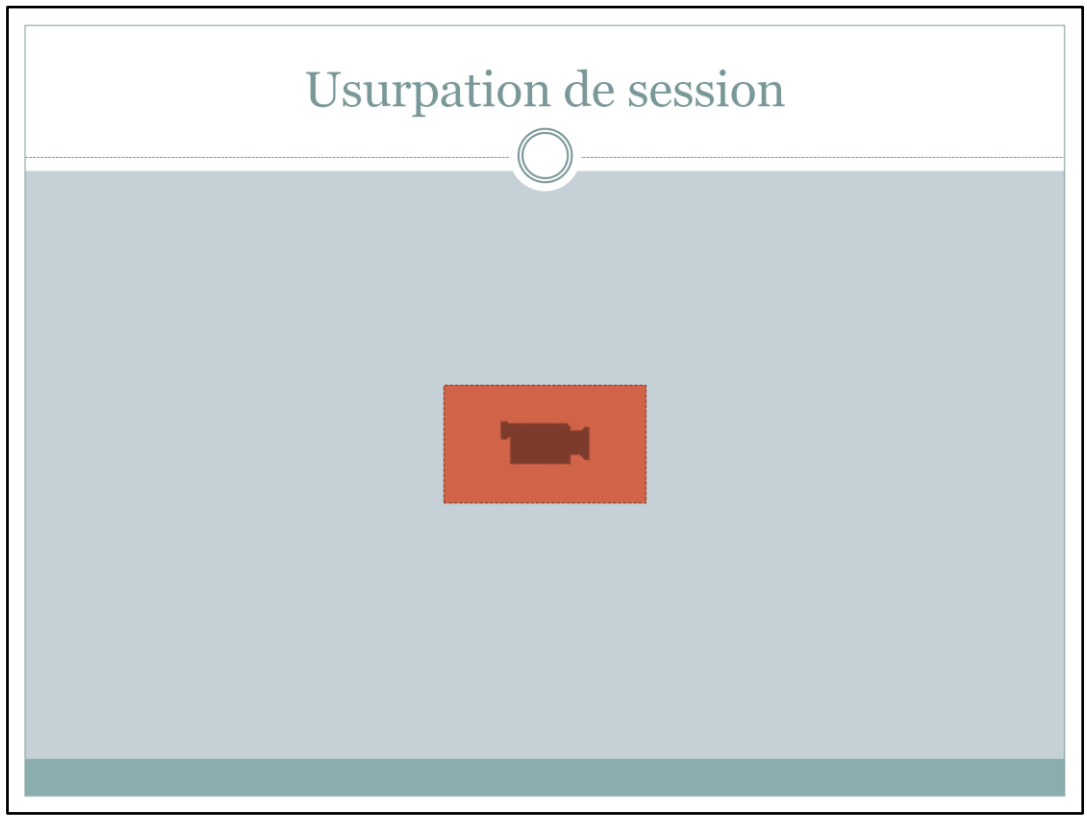
Voici la classe ctype qui pourra vous aider dans cette tâche :
http://fr2.php.net/manual/fr/ref.ctype.php

Usurpation de session

- En quoi ça consiste ?
  - L'usurpation de session consiste à accéder à la session d'un autre utilisateur
  - On distingue 3 types d'attaques de session :
    - Interception
    - Prédiction
    - Fixation
- Les risques
  - Un utilisateur malveillant peut prendre l'identité d'une autre personne, casser le site, envoyer des messages sous une autre identité, …

Usurpation de session

- Prédiction
  - Consiste à deviner un identifiant de session valide
- Interception
  - Consiste à récupérer l'ID de session (PHPSESSID) via une faille XSS comme tout à l'heure, ou via GET quand la configuration de PHP le permet
- Fixation
  - Consiste à fixer l'ID de session AVANT d'utiliser l'application
    - Le pirate envoie l'URL http://imp.free.fr/?Horde=1234 à un utilisateur
    - L'utilisateur suit le lien, s'il ne possède pas de cookie de session, il va initialiser la session à 1234
    - L'utilisateur entre son login et s'identifie
    - Le pirate suit alors le même lien http://imp.free.fr/?Horde=1234 et se retrouve à la racine du site, avec la session de l'utilisateur fraichement authentifié



Webmail de FREE

• La directive ini_set('session.use_trans_sid', '0') permet de bloquer le passage de l'ID de session d'un utilisateur par $_GET, c'est-à-dire par l'URL. Même si cette id de session est une chaîne aléatoire, le fait qu'elle passe en clair en facilite la récupération.

Le passage par l'URL de l'ID de session intervient généralement quand un utilisateur bloque au niveau de son navigateur tous les cookies des sites. En effet, quand un navigateur ne peut pas écrire votre ID de session dans un fichier, il essaie automatiquement de la passer par l'URL. Vous pouvez tester avec le webmail de Free. Pour bloquer les cookies, dans Firefox (Windwos), allez dans :

Outils -> Options… -> Vie privée et décochez « Accepter les cookies »

Les cookies sont de petits fichiers stockés par votre navigateur. Ils contiennent généralement des informations sur votre session en cours sur un site web. Ils peuvent aussi servir d'espion (spyware).

• Le script qui suit permet de sérieusement compliquer la vie du pirate :

La fonction session_name permet de changer le nom de la session par défaut qui est PHPSESSID et que vous avez pu voir dans les URL de la diapo précédente. C'est faciliter la vie du hacker que de laisser des options ou des noms par défaut! Sur le webmail de Free, cette session s'appelle « Horde ».

Ensuite, vous connaissez la fonction session_start, qui permet de démarrer un session unique sur votre site.

Ensuite, pour encore rendre la vie aux pirates un peu plus dure, on va régénérer un ID de session. On commence d'abord par tester l'existence d'une variable booléenne « session initiée », pour éviter de régénérer une session à chaque chargement de page, et, s'il elle n'existe pas, on régénère un ID de session et on crée notre variable booléenne citée précédemment.

Include / Require

- Pourquoi est-ce potentiellement dangereux ?
  - Si vous vous basez sur les tableaux $_GET et $_POST pour inclure des fichiers dans vos pages et que vous ne vérifiez rien, vous prenez le risque qu'un utilisateur mal intentionné vous vole des fichiers personnels (ou pire, des mots de passe)

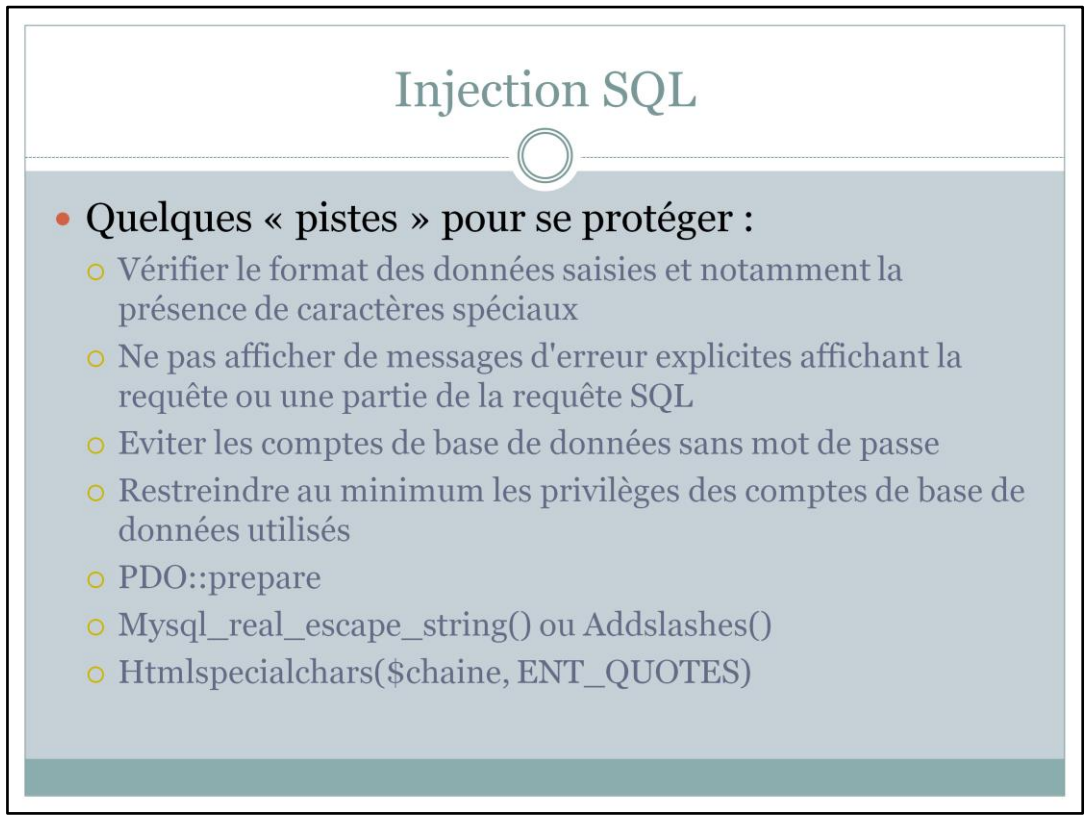
Include / Require

Include / Require

- Quelques pistes pour se protéger
  - Suivre les conseils du TD2
    - Gérer une liste blanche de pages que l'on peut « inclure »
    - Ne pas faire directement « include($_GET['page']) » mais plutôt « include("page_".$_GET['page']."html) » et nommer ses pages en conséquence

Injection SQL

- En quoi ça consiste ?
  - L'injection SQL consiste à exploiter une faiblesse d'un site web qui contrôle mal les données insérées par les utilisateurs (j'insiste)
- Les risques
  - Un utilisateur malveillant peut insérer du code SQL et :
    - Au pire, détruire ou modifier la base de données
    - Au mieux, accéder à des données confidentielles



•Comme préconisé dans le TD, utilisez PDO::prepare suivi de PDOStatement->execute. Cela vous protègera contre les injections SQL (n'utilisez pas directement PDO:: query sauf si votre requête SQL ne contient pas de variable).

•Les fonctions PHP mysql_real_escape_string (http://fr.php.net/manual/fr/function.mysql-real-escape-string.php) et addslashes (http://fr2.php.net/manual/fr/function.addslashes.php) permettent comme PDO::prepare d'ajouter des caractères d'échappement, i.e. « \ » devant les caractères réservés.

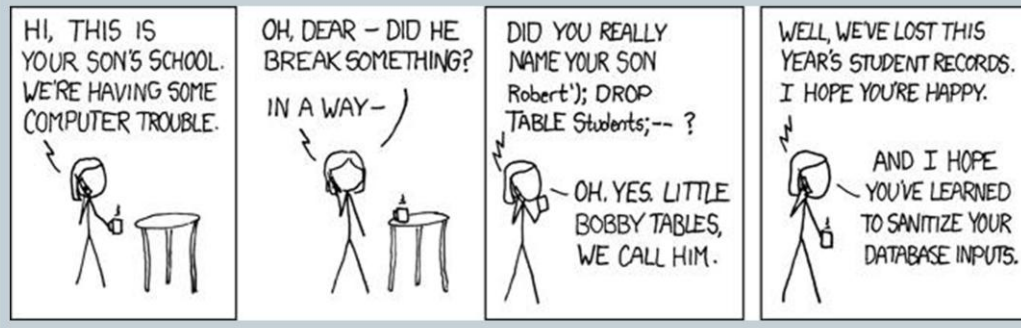
Si vous utilisez une base mysql, je vous conseille d'utiliser la fonction dédiée mysql_real_escape_string.

Sinon, addslashes fonctionnera aussi mais rajoutera plus de « \ » devant les caractères spéciaux. La fonction qui fait l'inverse (i.e. retirer tous les « \ ») s'appelle stripslashes (http://fr2.php.net/manual/fr/function.stripslashes.php).

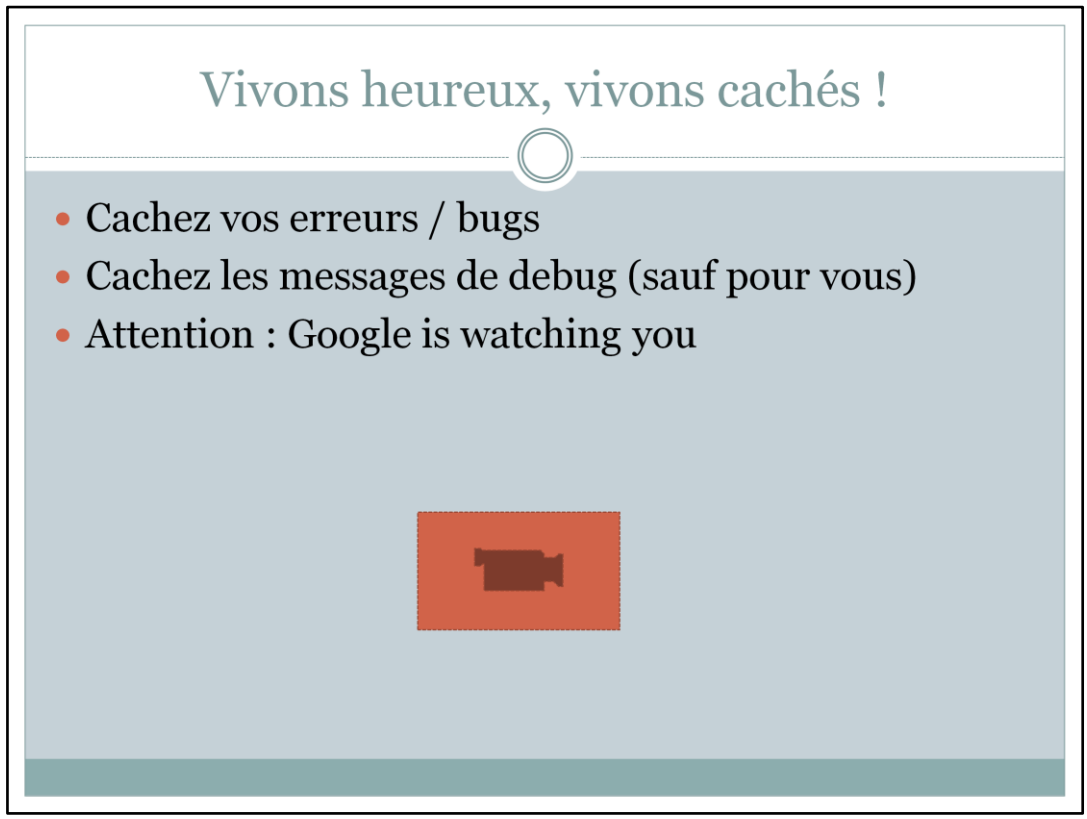
• Les messages de debug sont à bannir d'un site en production… en affichant l'erreur SQL avec la requêtes, vous montrez au pirate que vous ne contrôlez pas vos champs, et, cerise sur la gâteau, vous lui donnez toutes les informations pour bien malmener votre site !

• Essayez de restreindre les droits du compte utilisateur de base de données que vous utilisez dans votre code une fois votre site en production. Un utilisateur « admin » détourné pourra casser toute la base de données, et utiliser comme bon lui semble votre serveur. En production, un utilisateur avec les droits SELECT, INSERT, UPDATE, DELETE suffit généralement.

Injection SQL



A méditer ;)



• il n'est pas rare de trouver, en plus des messages de debug, la présence de la fonction phpinfo() qui permet généralement de tester votre configuration PHP. Cette fonction est une mine d'information…
• si vous avez oublié des pages privées sur votre site ou s'il y a une faille, comptez sur Google pour vous aider à les retrouver ! Attention, le Google Hacking est à utiliser avec parcimonie sinon Google se fâche et se bloque… Et quand cela arrive, c'est tous vos camarades que vous privez de Google… Et la DSI sait vous retrouver ;)

Contrôlez !

- Il ne faut pas faire confiance aux utilisateurs !
- Contrôlez toutes les données que l'utilisateur (gentil ou non) peut insérer :
  - $_GET
  - $_POST
  - $_COOKIE
  - $_REQUEST
- Si vous attendez un entier, vérifiez que vous récupérez un entier :
  - ctype_digit($var)
  - ctype_alpha($var)
  - etc.

Contrôlez !

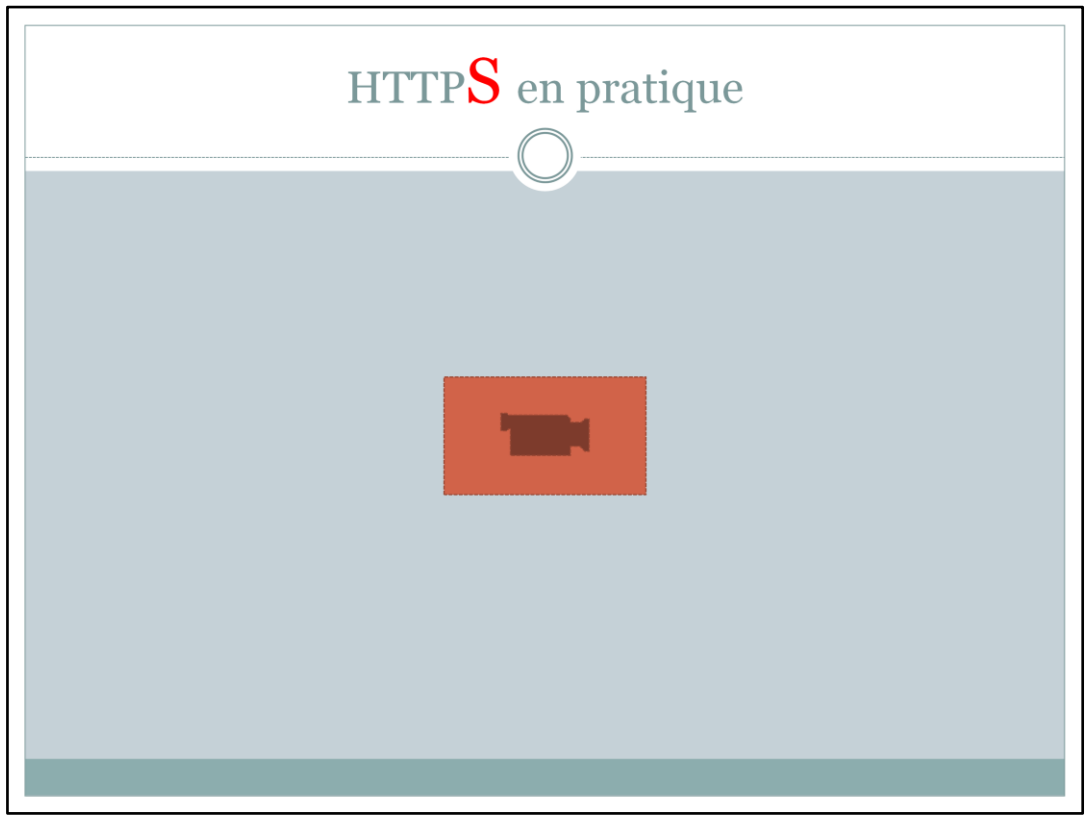
- Pour les données plus évoluées, utilisez les expressions régulières (REGEX) :

```
$nom_ou_prenom = "jéan de l'a lune";

// ==FALSE si $nom_ou_prenom est NULL ou si ne contient pas au moins une lettre,
// puis optionnellement des groupes commençant par un séparateur (espace, ' ou -) avec au moins une lettre derrière

$ok=($nom_ou_prenom==NULL || preg_match('#^[[:alpha:]]+([\-\' ][[:alpha:]]+)*$#', $nom_ou_prenom));
```

- Pimentez un peu les mots de passe bidons :
  - $clef_du_site = 'kjhgfskdjfgç_è-a'lkj';
  - $password_securise = $_POST['password'].$clef_du_site;
  - INSERT INTO utilisateurs (login,password) ('$login',SHA1('$password_securise'))



Captures wireshark HTTP / HTTPS

Conclusion

- Contrôlez !!!
- Sécurisez les données sensibles avec HTTPS (si possible)
- Méfiez-vous de ce que vous trouvez sur Internet
  - PhpMyAdmin : 8 failles de sécurité en 2010
  - Mediawiki : 4 failles de sécurité en 2010
  - PhpMyVisites : 1 faille critique fin 2009 permettant de prendre le contrôle du serveur !
  - ...
- Payant ne veut pas dire plus sécurisé !
- La plupart des intrusions sont faites de manière automatiques par des robots (GoogleHacking, ...)

•<mode parano on> Ne faites jamais confiance aux utilisateurs !!! <mode parano off>

Quelques liens

- http://phpsec.org/projects/guide/fr/index.html
- http://fr.php.net/register_globals
- http://www.phpfreaks.com/tutorial/php-security/page1
- http://code.google.com/p/browsersec/wiki/Main
- http://johnny.ihackstuff.com/ghdb.php
- http://julien-pauli.developpez.com/tutoriels/securite/developpement-web-securite/
- http://www.zytrax.com/tech/web/regex.htm