

Tuto JRES 2010

Conseils en écriture PHP/MySQL

Magali Contensin

contensin@ibdml.univ-mrs.fr

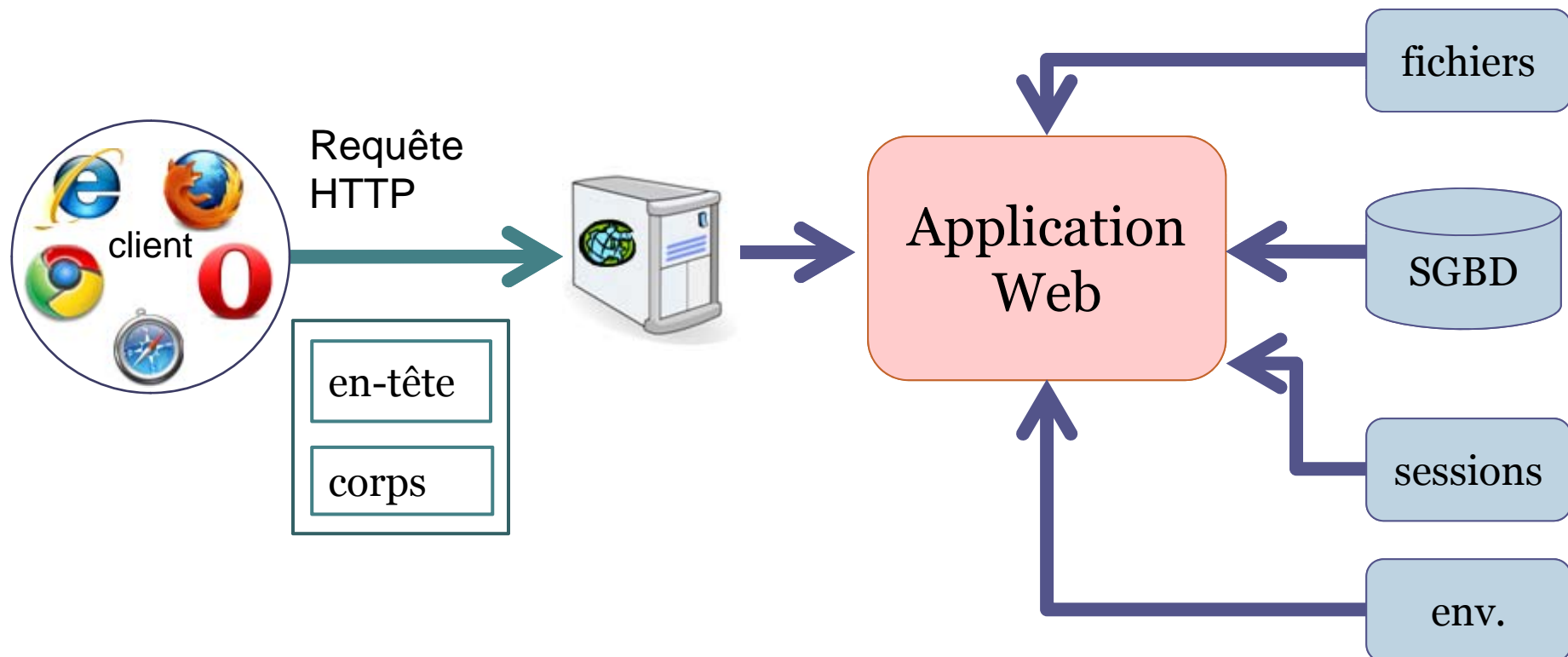


Plan

1. Filtrer les entrées
2. Protéger les sorties
3. Interdire la mise en cache de données
4. Masquer les données de connexion à un SGBD
5. Protection contre le XSS
6. Protection contre les Injections
7. Protection contre le CSRF
8. Protection contre le détournement de sessions

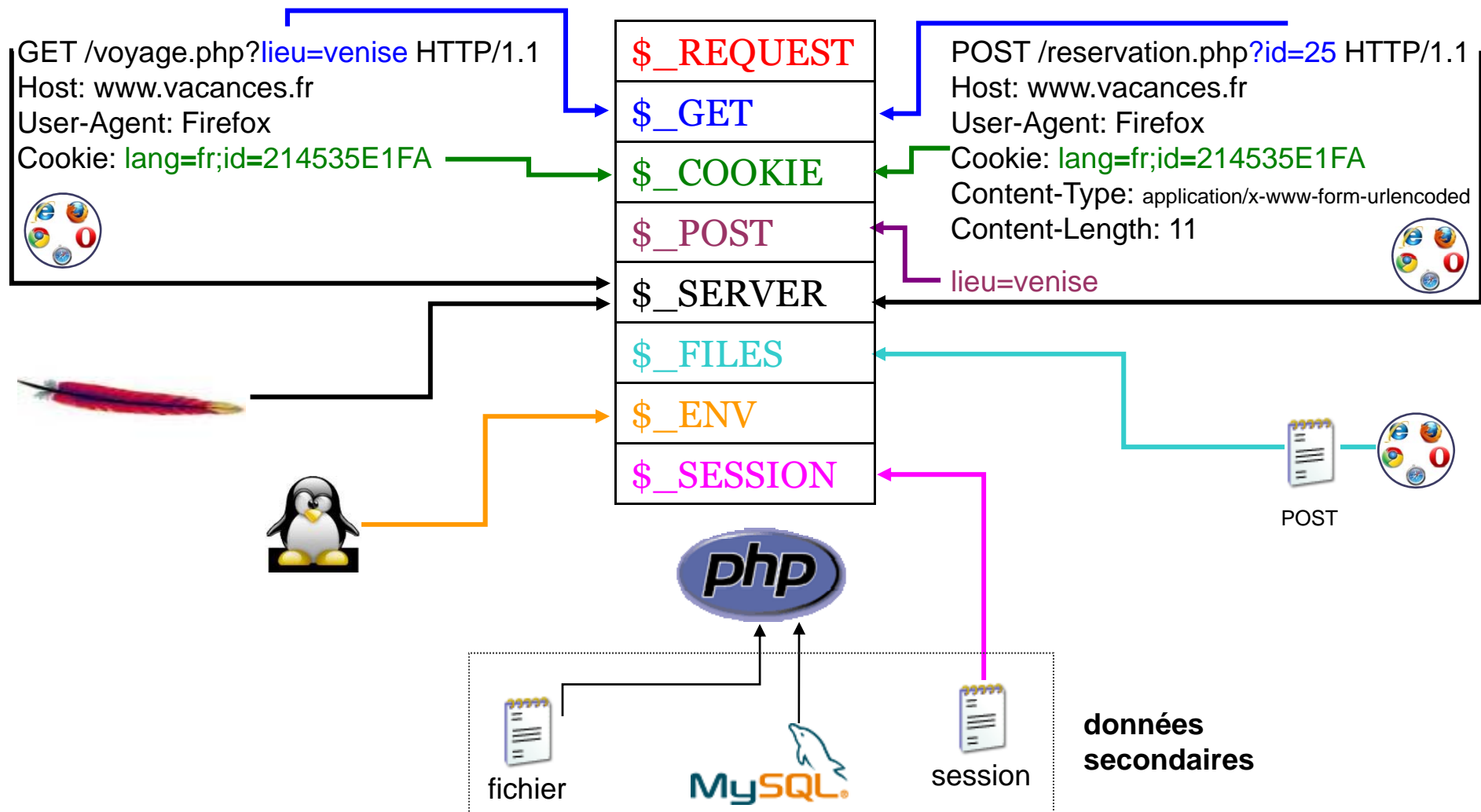
1. Filtrer les entrées

- Entrées d'une application web



1. Filtrer les entrées

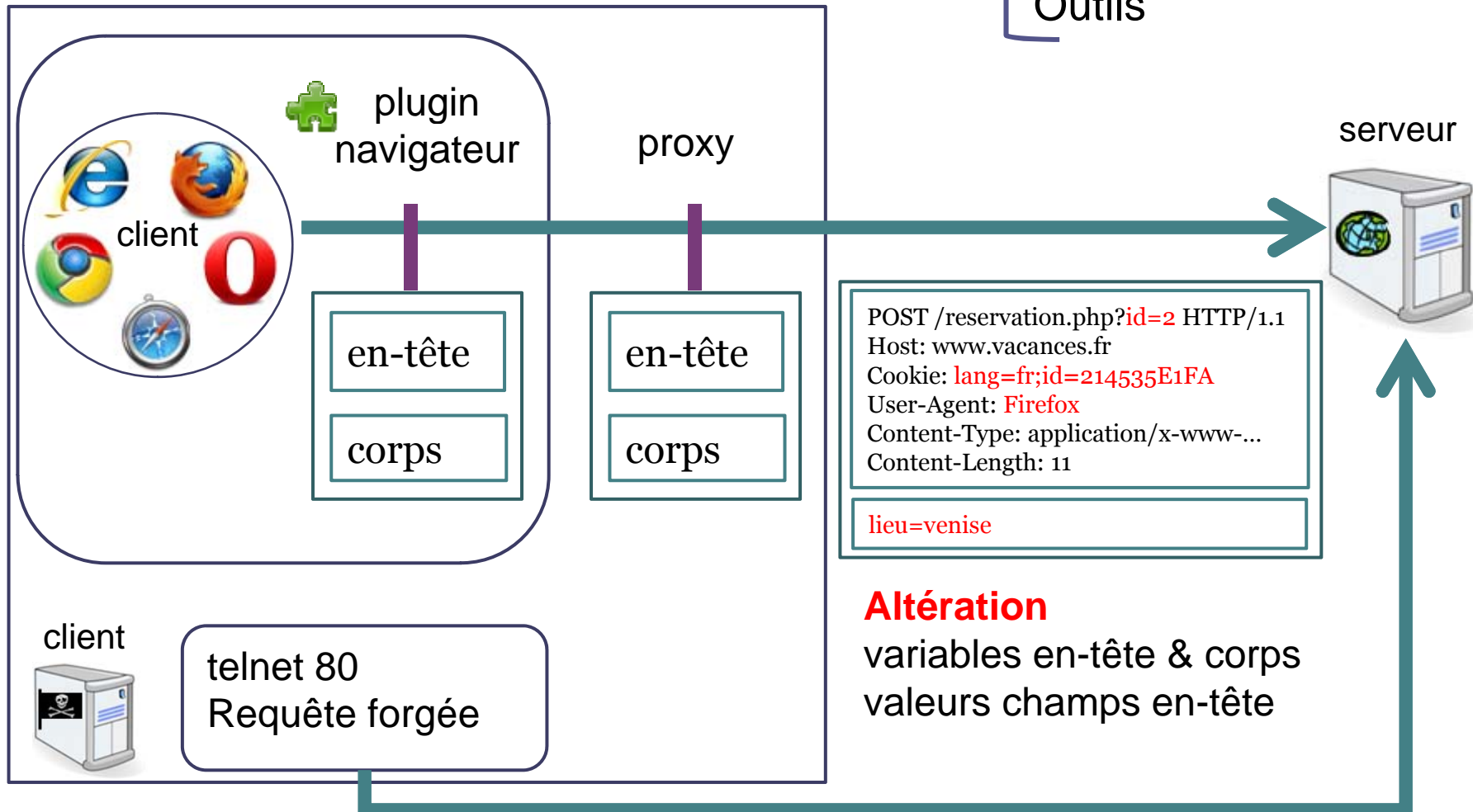
■ Entrées d'un script PHP pour GET / POST



1. Filtrer les entrées

■ Envoi de données malveillantes

Query string
Formulaire
Requête forgée
Outils



1. Filtrer les entrées

■ Implications

- méthode HTTP différente de celle attendue
- champs en-tête HTTP modifiés (User-Agent)
- arguments supprimés ou modifiés (valeurs inattendues)
- arguments ajoutés

```
<?php // login.php
...
if (verifLogin($_POST['login'], $_POST['passwd'])){
    $ok = true ;
}
...
if ($ok){
    ...
}
?>
```

<http://.../login.php?ok=1>

extract
register_globals à on

1. Filtrer les entrées

- Données reçues peuvent être :
 - retournées au client
 - ↳ XSS réfléchi
 - stockées dans SGBD, fichier, session
 - ↳ XSS stocké
 - utilisées pour interroger un SGBD
 - ↳ SQL Injection
 - utilisées pour exécuter un programme
 - ↳ Injection de commandes

Ne jamais faire confiance aux données envoyées par le client

1. Filtrer les entrées

- Initialiser les variables

- Vérifier toute entrée

`$_GET`, `$_POST`, `$_COOKIE`, `$_REQUEST`, `$_FILES`, `$_SERVER`,
`$_ENV` et `$_SESSION`

- Contrôles à effectuer

- type des données : `intval`, `ctype_*`, `cast`
- présence des données attendues
- valeurs de nombres comprises entre deux bornes
- `tailleMin < taille donnée < tailleMax`
- jeu de caractères
- valeur null autorisée ou non
- `select`, `radio`, ... : valeur comprise dans une liste

1. Filtrer les entrées

■ Filtrage

- liste blanche :

 - Indique ce qui est autorisé

- liste noire :

 - Indique ce qui est interdit

 - A éviter car plus permissif

Une extension PHP (version > 5.2) fournit des fonctions de filtrage et de nettoyage (fonctions `filter_*`)

1. Filtrer les entrées

■ Protéger les opérations sur les fichiers

(file, readfile, unlink, include, upload, ...)

- Vérification du chemin `realpath($chemin)`
- Suppression du chemin `basename($chemin)`

```
<?php // script code.php
header("Content-type:text/plain");
$fichier = basename($_GET["p"]);
readfile($fichier);
?>
```

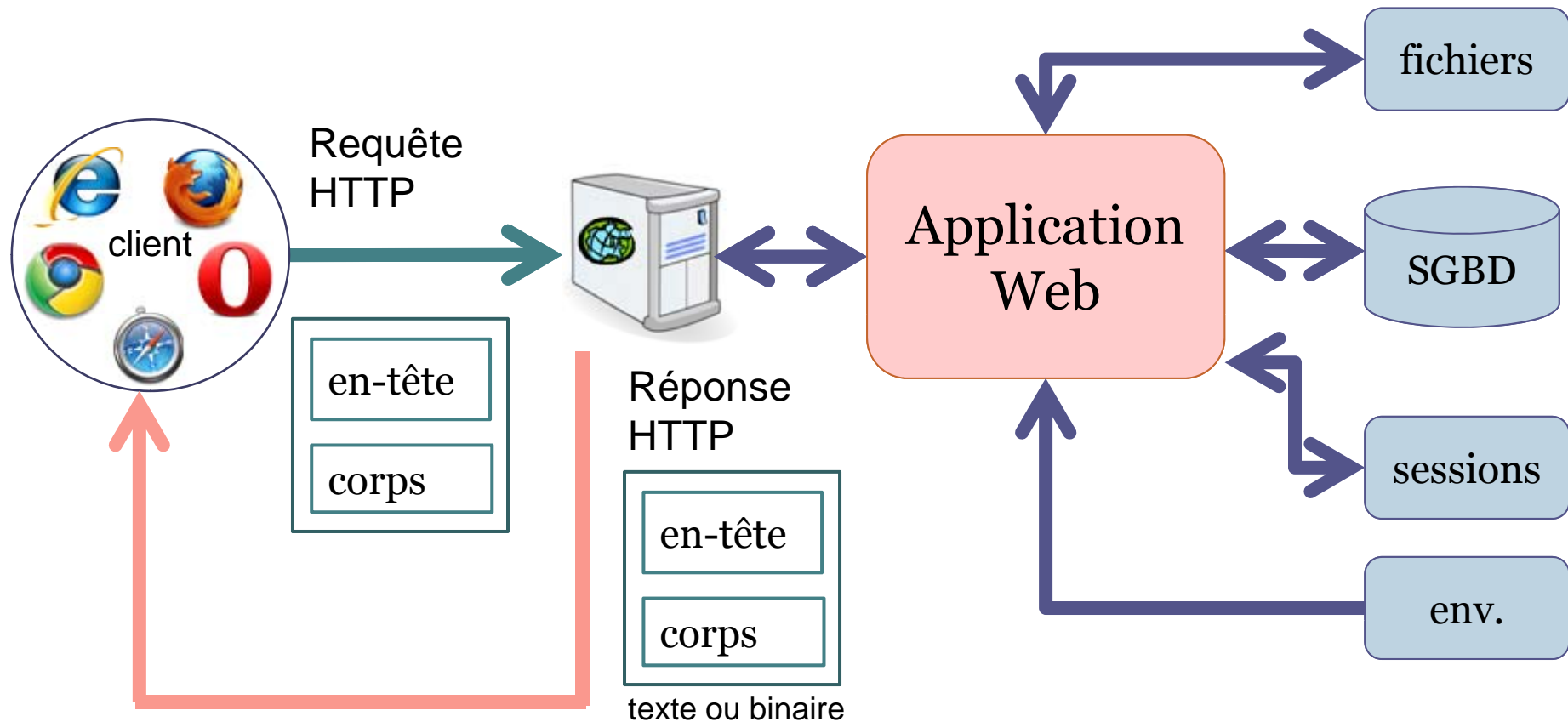
```
http://...../code.php?p=/etc/passwd
```

1. Filtrer les entrées

- Protéger les opérations sur les fichiers
 - File upload
 - Lire les données avec `$_FILES`
 - Manipuler avec `move_uploaded_file`
 - Ne pas se fier au type MIME ou à l'extension
 - Attention aux commentaires d'images
 - Renommer (ne jamais utiliser le nom de fichier envoyé)
 - Limiter la taille et le nombre des téléchargements.

2. Protéger les sorties

- Sorties d'une application web



2. Protéger les sorties

■ Navigateur

- Définir le jeu de caractères de la page

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
```

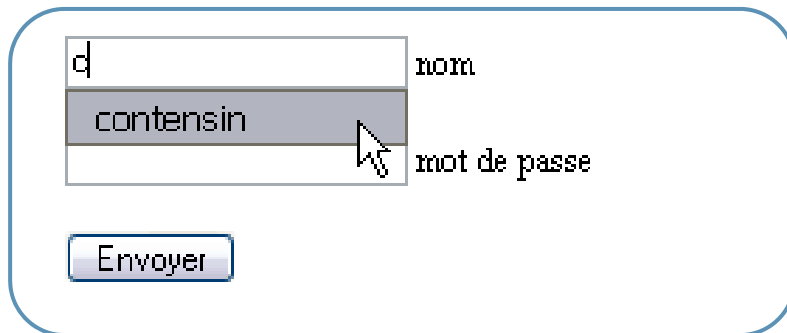
- Coder les caractères spéciaux

```
htmlspecialchars, htmlspecialchars(chaine, ENT_QUOTES)
```

■ Données passées à un interpréteur

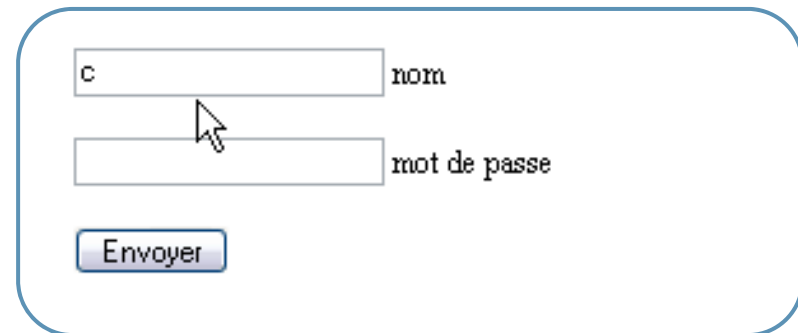
3. Interdire la mise en cache

■ Complétion de formulaires



nom
d
contensin
mot de passe
Envoyer

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



nom
c
mot de passe
Envoyer


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

■ Page web

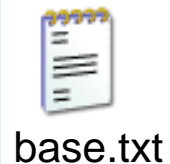
```
header("Cache-Control: no-cache"); // HTTP/1.1  
header("Pragma: no-cache"); // HTTP/1.0  
header("Expires: Thu, 01 Jan 1970 00:00:00 GMT");
```

4. Masquer les informations de connexion SGBD

```
<Directory "/var/www/PagesPerso/magali/appliWeb">  
    Include /home/magali/base.txt  
</Directory>
```



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



```
chmod 600 base.txt
```

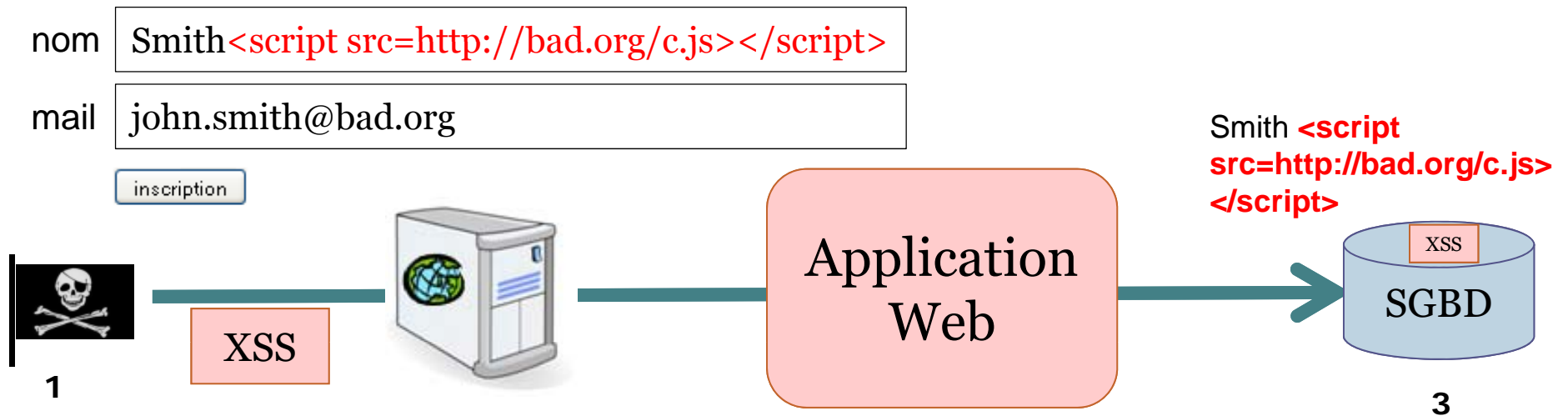


```
<?php  
    echo $_SERVER['dbLogin'];  
    system('cat /home/magali/base.txt');  
?>
```



5. Protection contre le XSS

- XSS envoyé par le pirate



Erreur 1 :
pas de vérification
des entrées

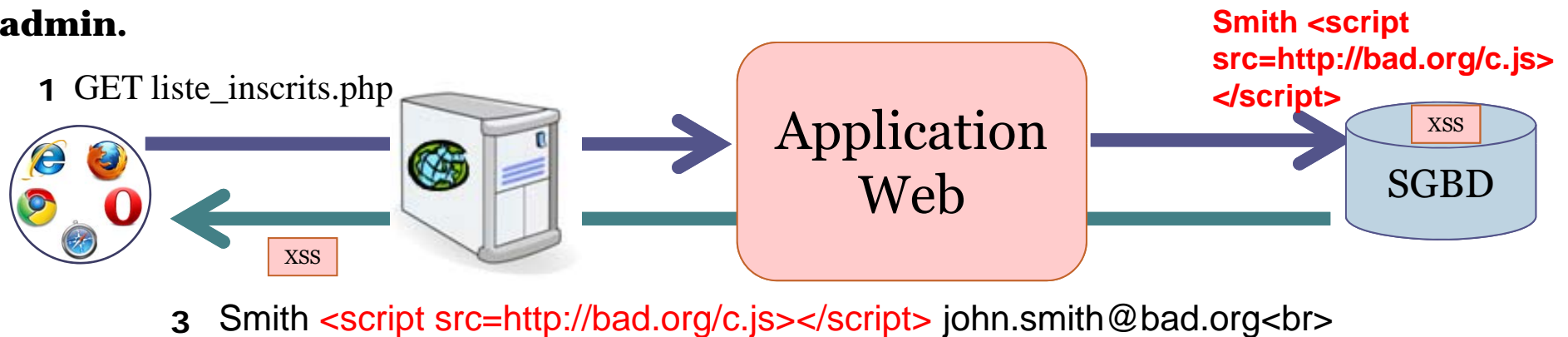
2

```
<?php
import_request_variables('g', 'f_');
$req = "INSERT INTO users (nom, mail)
      VALUES ('$f_nom', '$f_mail')";
...
?>
```

5. Protection contre le XSS

- L'accès à une ressource provoque l'envoi du XSS au navigateur => vulnérabilité

admin.



Erreur 2 :
pas de protection
des sorties

2

```
<?php
$req = "SELECT nom, mail FROM users";
...
echo $res['nom'];
?>
```

5. Protection contre le XSS

- Filtrer les entrées (par liste blanche)
éventuellement nettoyer les entrées
(`utf8_decode`, `strip_tags`, `filter_*`)
- Préciser le jeu de caractères
- Protéger les sorties (`htmlspecialchars`, `htmlspecialchars`)

```
<?php
$req = "SELECT nom, mail FROM users";
...
echo htmlspecialchars($res['nom'], ENT_QUOTES);
?>
```

Smith `<script src=http://bad.org/c.js></script>`

6. Protection contre les injections

■ SQL



login mot de passe

1

```
POST http://.../cnx.php HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 24

login=%27+OR+1+=+1+--+%20
```

Application
Web

2

```
<?php
...
$login = $_POST['login'];
$pwd = $_POST['passwd'];
$req = "SELECT login, prenom FROM users
      WHERE login='$login' AND password='$pwd'";
...
// $prenom contient la valeur retournee par MySQL
echo "bonjour $prenom !";
?>
```

6. Protection contre les injections

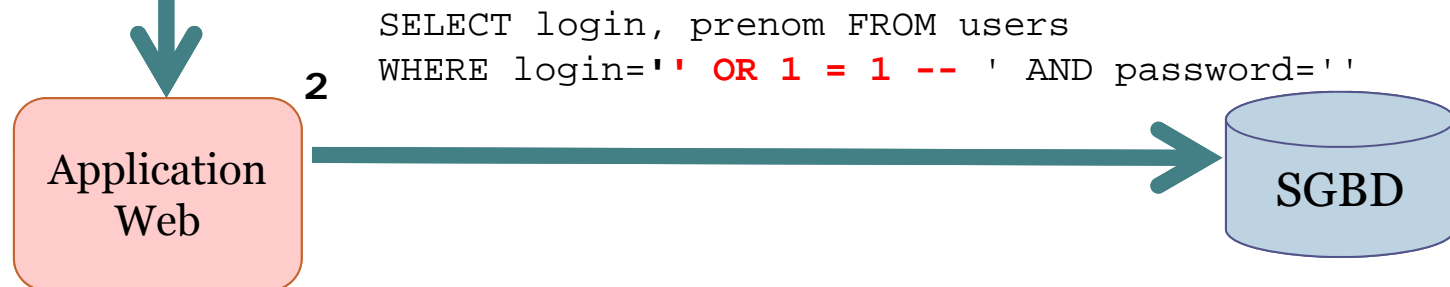
■ SQL



login mot de passe

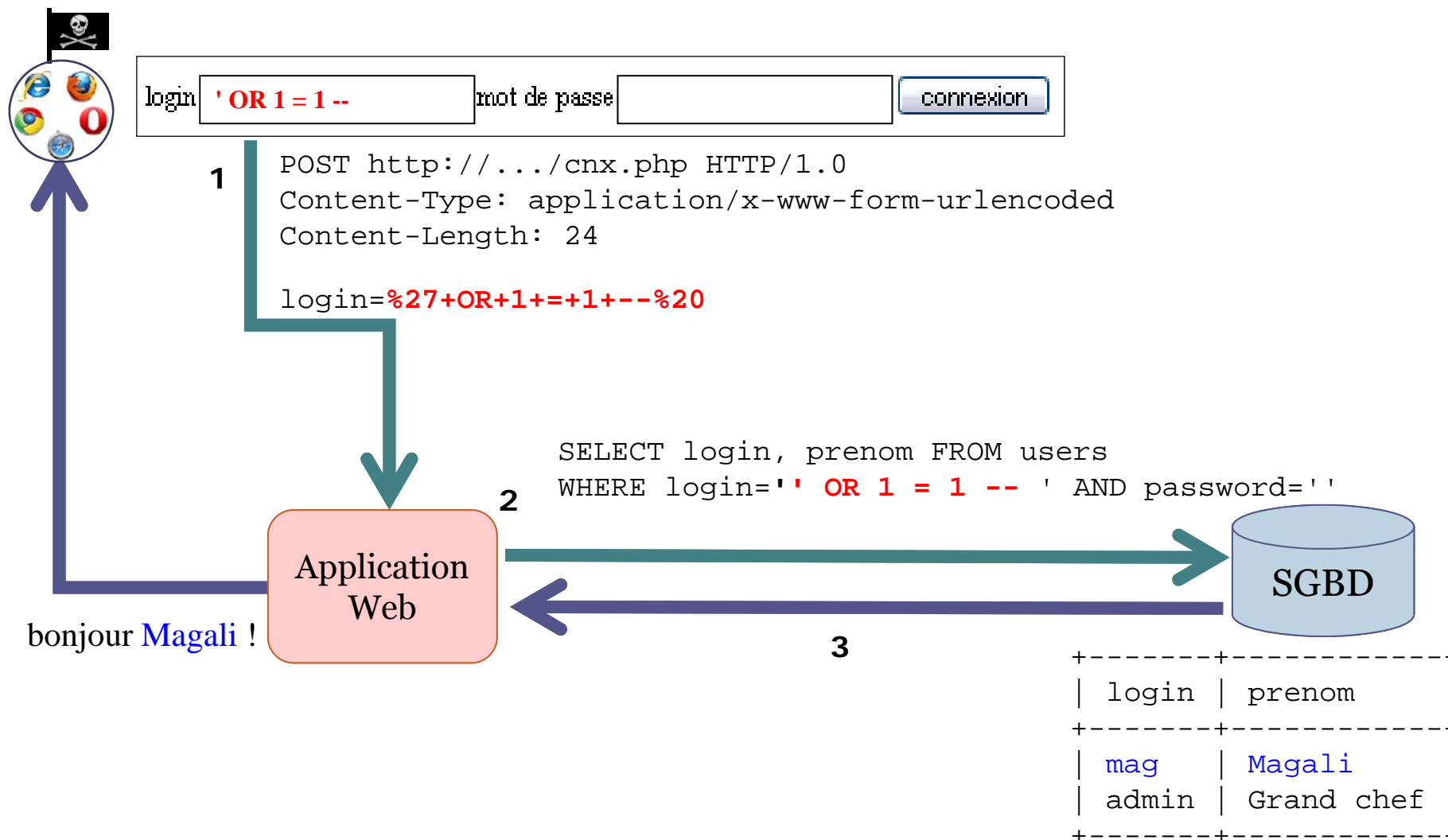
1
POST http://.../cnx.php HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 24

login=%27+OR+1+=+1+--%20



6. Protection contre les injections

■ SQL



6. Protection contre les injections

- Filtrer les entrées : taille, type, ...
- Cast des nombres ou mettre des ' ' autour des nombres

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

```
?ref=21&nb=5,%20desc=LOAD%20INFILE('/etc/passwd')
```

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

- Ne pas afficher d'information en cas d'erreur
- Extension mysql : protéger avec mysql_real_escape_string

```
$req = "SELECT login, prenom FROM users
WHERE login='" . mysql_real_escape_string($login) .
"' AND password='" . mysql_real_escape_string($pwd) . "'";
```

login	<input type="text" value="' OR 1 = 1 LIMIT 1,1 --"/>	mot de passe	<input type="text"/>	<input type="button" value="connexion"/>
-------	--	--------------	----------------------	--

```
SELECT login, prenom FROM users WHERE login='\ ' OR 1 = \1 LIMIT 1,1 -- ' AND password="";
```

6. Protection contre les injections

- Utiliser des requêtes préparées avec des paramètres liés (sinon aucune protection)

```
<?php
require 'db.php';
try{
    // connexion
    $cnx = new PDO("mysql:host=$host;dbname=$db_name", $db_user, $db_pwd);
    $cnx->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // preparer la requete
    $req = "SELECT prenom FROM users WHERE login=? AND password=?";
    $reqprep = $cnx->prepare($req);
    $reqprep->execute(array($_POST['login'], $_POST['passwd']));
    // afficher le resultat
    if ($res = $reqprep->fetch(PDO::FETCH_ASSOC)){
        echo "bonjour ", $res['prenom'];
    }
    // deconnexion
    $cnx = null;
}catch (PDOException $e){
    die("exception");
}
?>
```

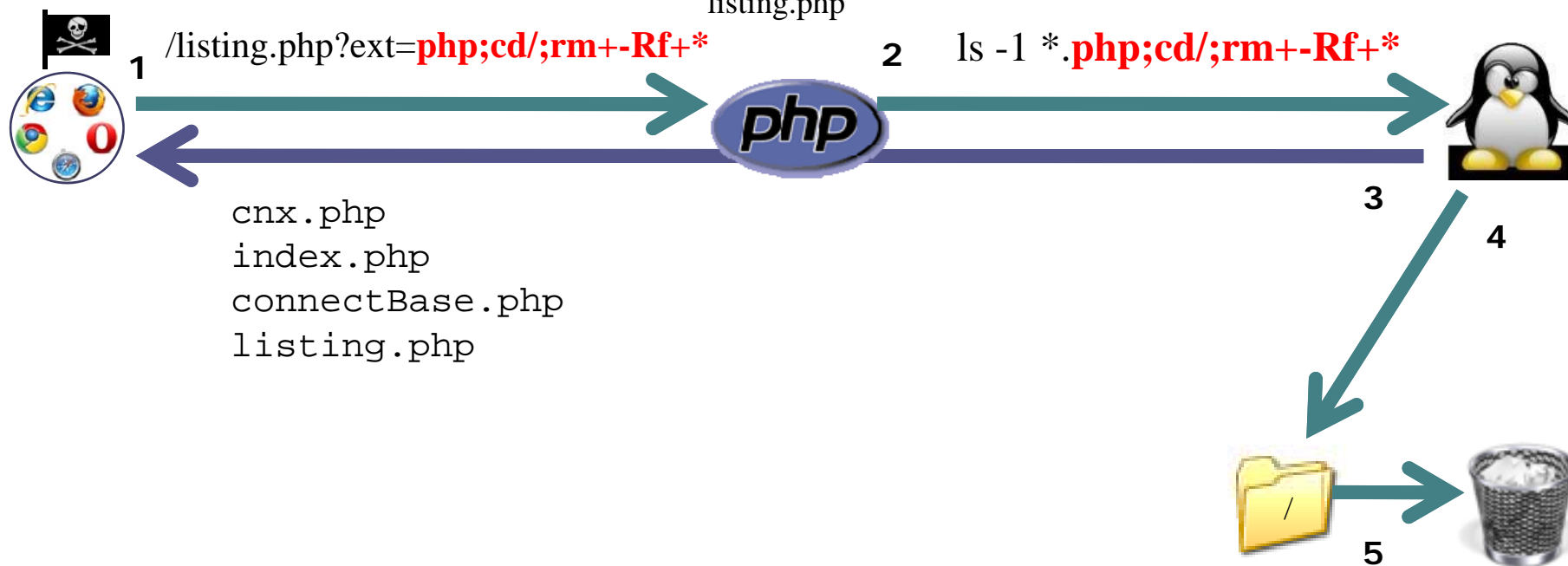
6. Protection contre les injections

■ Commandes

```
<?php
echo system("ls -1 *.".$_GET['ext']);
?>
```



listing.php



6. Protection contre les injections

- Filtrer les entrées

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

- Supprimer tout appel système inutile :

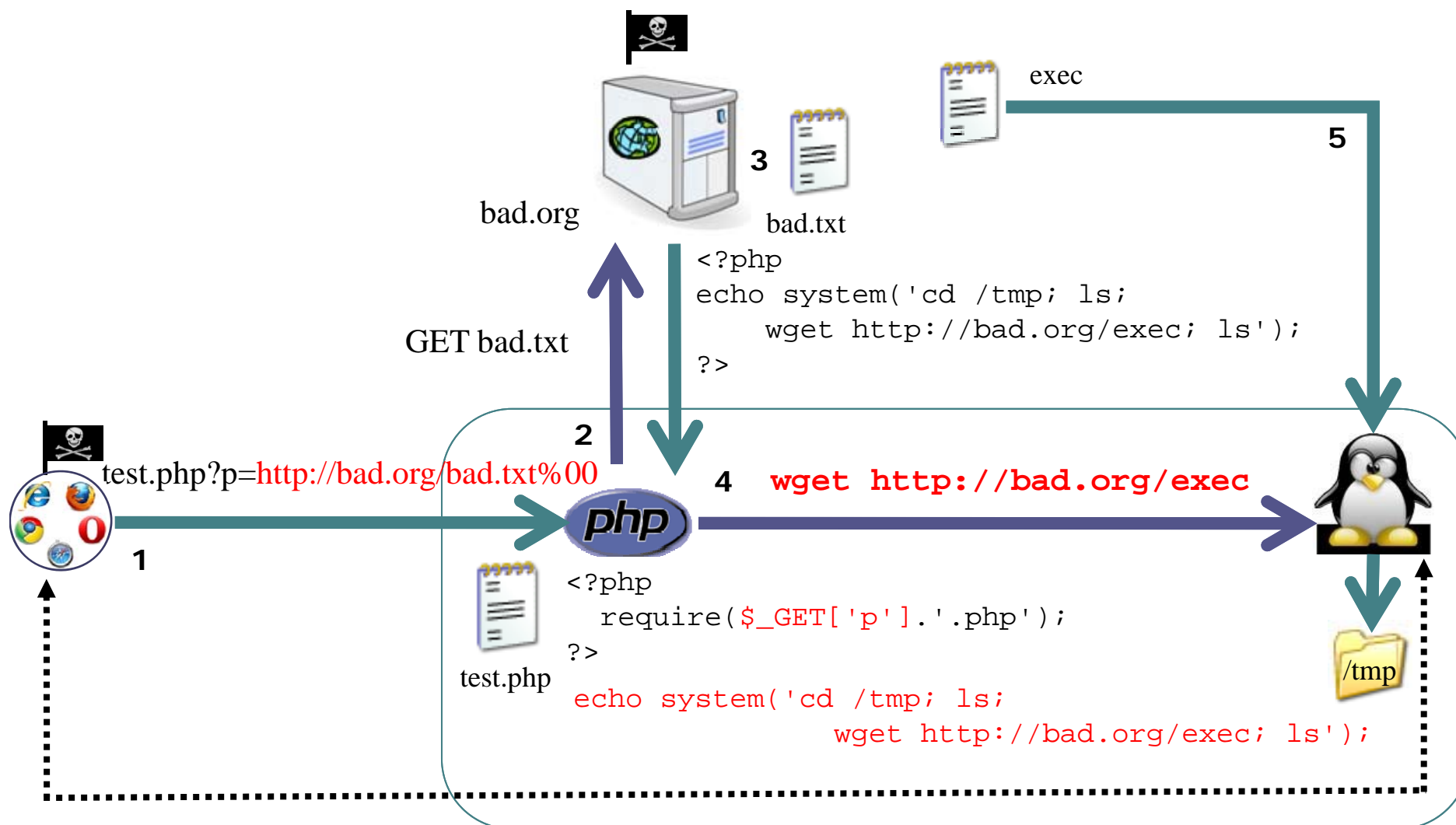
Ex. `system('sendmail...')` => `mail(...)`

- Protéger des méta-caractères

(`escapeshellcmd`, `escapeshellarg`)

6. Protection contre les injections

■ Code (RFI – Remote File Inclusion)



6. Protection contre les injections

- Filtrer les entrées (liste blanche)
- Chemin : utiliser basename, realpath

6. Protection contre les injections

■ Null injection (caractère \0)

```
<?php
    $fichier = basename($_GET['nom']);
    readfile($fichier.".txt");
?>
```

?nom=test.php%00

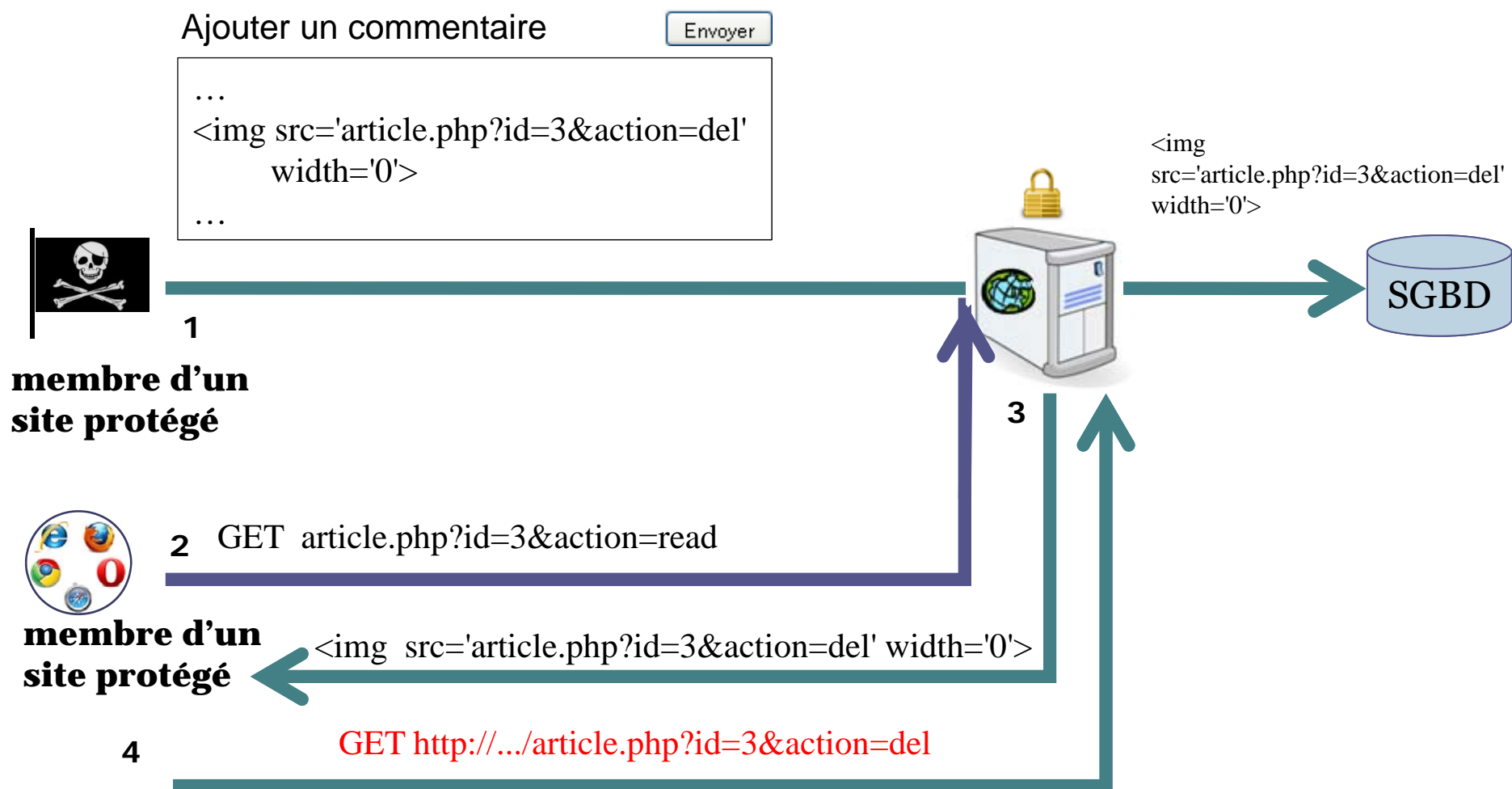
```
<?php
    if (preg_match("/^[^A-Za-z]/", $_GET['chaine'])){
        die('erreur');
    } else{
        echo 'ok';
    }
?>
```

?chaine=test1

?chaine=%00test1

Filtrer les entrées (méthode liste blanche)

7. Protection contre le CSRF



7. Protection contre le CSRF

■ Actions à entreprendre

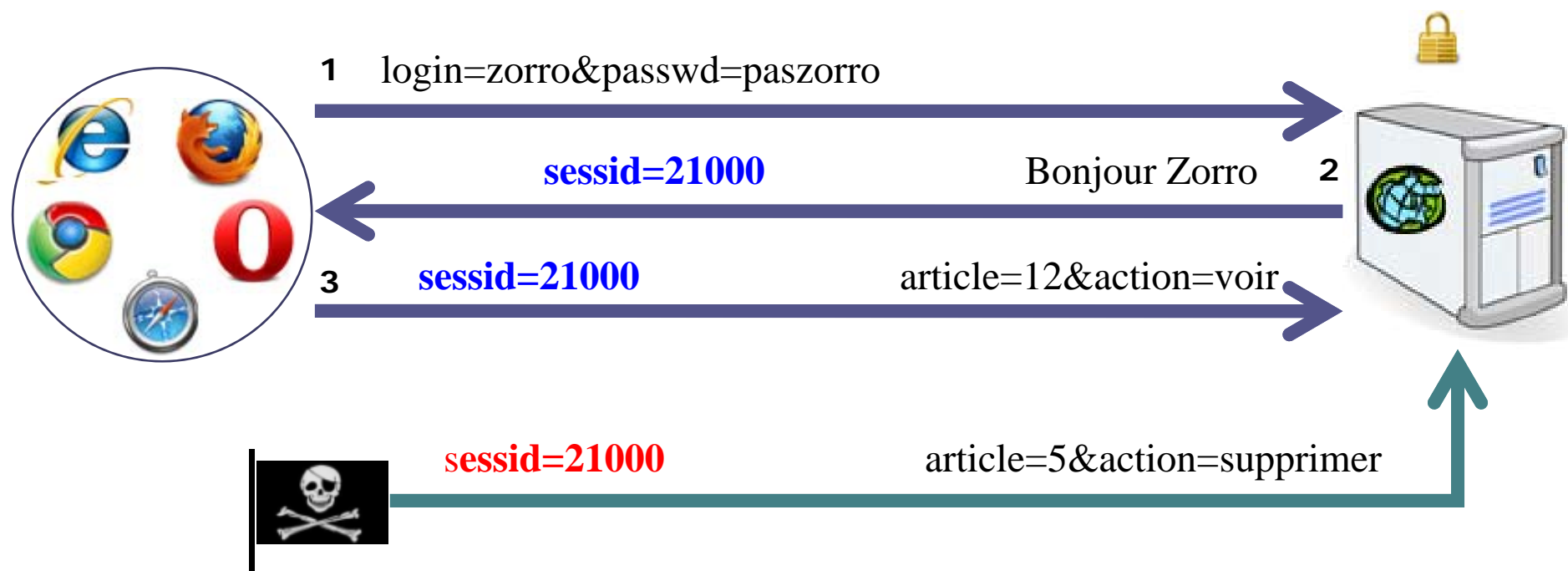
- Majorité des attaques = méthode GET
 - Utiliser POST si mise à jour de données
 - Ne pas utiliser \$_REQUEST pour récupérer les données

=> Protège contre les attaques menées par GET
- Attaques utilisant POST
 - Forcer l'utilisation du formulaire du site (idée s'assurer que les données reçues par POST proviennent bien d'un formulaire envoyé par le site)

=> Utiliser un identifiant aléatoire pour chaque formulaire, stocké en session avec un délai de validité

8. Protection contre le vol de session

- Détourner une session = fournir un id valide



8. Protection contre le vol de session

- Identifiant transmis par cookie, URL, champ form.
- Attaques pour obtenir un identifiant valide
 - Prédiction (id = nombre entier incrémenté)
 - Force brute
 - Fixation
 - Vol
 - id dans l'URL (historique, bookmark, logs, envoi par mail, ...)
 - vol de cookie (XSS, ordinateur public)
 - interception (écoute réseau)
 - consultation des fichiers de session sur le serveur

8. Protection contre le vol de session

- Régénérer l'id de session après authentification (session_regenerate_id)
- Limiter la durée de validité d'une session
- Utiliser un 2^{ème} moyen de suivi de session (chaîne User-Agent encodée, stockée en session)
- Toujours proposer une déconnexion, celle-ci doit :
 - ❑ Supprimer les données de session sur le serveur
 - ❑ Envoyer un cookie de session avec une valeur vide au client
- Utiliser les identifiants de session PHP
- Transmettre l'id uniquement par cookie

Sur Internet

- **Top 25 Most Dangerous Programming Errors**
<http://cwe.mitre.org/top25>
- **WASC** (Web Application Security Consortium)
http://www.webappsec.org/projects/threat/classes_of_attack.shtml
- **OWASP** (Open Web Application Security Project)
http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
http://www.owasp.org/index.php/Category:OWASP_Guide_Project
- **CERTA** (Centre d'Expertise Gouvernemental de Réponse et de Traitement des Attaques Informatiques)
<http://www.certa.ssi.gouv.fr>
- **CERT** (Computer Emergency Response Team)
<http://www.cert.org>