ITCE NFE102
Année 2012-2013

Problèmes informatiques du commerce électronique

F.-Y. Villemin (f-yv@cnam.fr)

http://dept25.cnam.fr/ITCE

Basé sur des textes de D. Gehl, A. Dulaunoy, N. Biot, Nexen

Introduction

La construction des serveurs web posent un certain nombre de problèmes récurrents :

- L'authentification du client (reconnaître un client, l'autoriser à certains services…)

  Le protocole HHTP 1.1 permet 2 modes d'authentification d'un client

- Les bases de données des applications existantes (BackOffice) ne sont faites pour être interroger par un client web, il faut les présenter sous la forme de catalogues navigables

- Dans les applications BtoB, les systèmes des clients et fournisseurs communiquent directement entre eux (sans intermédiaires humains), il doivent s'échanger des informations comprises de manière identique pour tous les acteurs, d'où des informations très structurées (cas des appels d'offre)

Authentification

L'authentification client est le processus de vérification de l'authenticité d'un client

Authentification HTTP/1.1

RFC2617 : Basic et Digest Access Authentification

L'authentification "Basic" est basée sur le principe le client doit s'authentifier pour chaque "realm" (domaine) ou répertoire

La notion de `realm` (domaine) permet de ne pas répéter la demande de mot de passe pour accéder à une ressource appartenant à la même région

les `login/passwd` sont stockés sur le poste client

Serveur Web : contrôle d'accès

Le "realm" est une valeur opaque :

```
realm="Mon Site"
```

La chaine d'authentification est limitée à 76 caractères :

```
Authorization: Basic <base64 login:passwd>
```

Exemple :

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

Schéma d'authentification `basic` (protocole HTTP)

- le client envoie un `nom_usager` et `mot_de_passe`
- le serveur répond :

```
HTTP/1.0  401  Unauthorized
WWW-Authenticate: Basic realm="Mon Site"
```

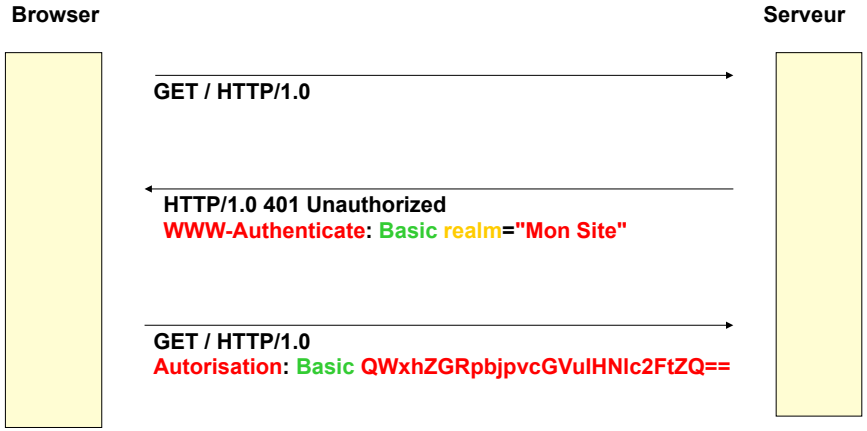
- le client demande `login` et `passwd` à l'usager
- le client uuencode `login:passwd` et le traduit en base64
- le client renvoie la requête avec l'en-tête :

```
Authorization: Basic xxxxxxxxxx
```

- si authentification correcte, le serveur renvoie la ressource, sinon retour à la deuxième étape

Authentification

Schéma d'authentification `basic` (protocole HTTP)



Exemple de HTTP Basic

Le client émet une requête :

```
GET /admin/ HTTP/1.1
```

Le serveur répond avec une entête (header)

```
WWW-Authenticate :
HTTP/1.1 401 Authorization Required
WWW-Authenticate: Basic realm="Mon Site"
```

Le client (suite à la saisie d'un password) émet une nouvelle requête avec :

```
GET /admin/ HTTP/1.1
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```


HTTP Digest

HTTP Digest n'est généralement pas implanté par les navigateurs

Le principe du HTTP Digest est que le client et le serveur partagent un secret : le client doit montrer qu'il connaît le secret

HTTP Digest est basé sur la notion de "message digest" : le calcul un résumé d'une chaîne avec une fonction H de hachage (MD5, SHA1…)

- Le serveur : ←`nonce` chaîne publique (plus ou moins aléatoire)
- Le client : `code = H(nonce:secret)`→
- Le serveur compare `code` reçu à `H(nonce:secret)`, si identique, cela montre que le client connaît le secret `secret` qui n'a pas circulé

Exemple de HTTP Digest

Le client émet une requête :

```
GET /admin/ HTTP/1.1
```

Le serveur répond avec une entête (header)

```
WWW-Authenticate:
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest
realm="Mon Site",
qop="auth",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

chaînes générées par le serveur à chaque réponse 401

Exemple de HTTP Digest

Le client répond (après le calcul de la réponse) :

```
Authorization: Digest username="moi",
realm="Mon Site",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="/admin/",
qop=auth,
```
niveaux de protection (`auth` ou `auth-int`)

`auth-int` : authentification avec vérification d'intégrité

```
nc=00000001,
cnonce="0a4f113b",
```
chaîne générée par le client

```
response="6629fae49393a05397450978507c4ef1",
opaque="5ccc069c403ebaf9f0171e9517f40e41"
```
chaîne générée par le serveur que le client doit retourner tel quel

Le serveur répond aussi une "reponse" pour le realm : `/admin/`

Authentification

HTTP Digest a été conçu pour résoudre plusieurs problèmes de HTTP Basic :

- L'interception du password,
- Le rejeu possible (replay attack),
- La vérification du client ET du serveur,
- Le stockage des password sur le serveur,
- La vérification de l'URI,...

L'authentification "Digest" est basée sur le principe "challenge/response". Le serveur envoie un challenge au client pour chaque "realm"

Une fonction de "hashage" est utilisée dans le cadre du "challenge/response" pour ne pas divulguer le secret partagé.

`Apache (mod_auth_digest)`

Droits d'accès HTTP avec Apache

Pour obliger les visiteurs à avoir nom de client et mot de passe pour le site :

- Le contrôle d'accès se fait au niveau du répertoire
- L'administrateur doit fournir au client son mot de passe, le client ne peut pas le modifier lui-même

Il suffit de placer un fichier .htaccess dans le répertoire à protéger
Le fichier `.htaccess` doit être en lecture pour tout le monde :

```
AuthName "Mon Site"
AuthType Basic
AuthUserFile /home/www2/clients/Login/HTML/.htpasswd
require valid-user
```

Il faut ensuite créer le fichier `.htpasswd` avec la commande `htpasswd`
Le fichier `.htpasswd` doit aussi être en lecture pour tout le monde :

```
$ htpasswd -c /home/www2/clients/Login/HTML/.htpasswd client
New password:
Re-type new password:
Adding password for user client
```


Accès à l'information en PHP

Les fonctions d'authentification HTTP de PHP ne sont disponibles que si PHP est exécuté comme module Apache

La fonction `Header()` pour utilisée pour demander une authentification ("`Authentication Required`") au client, générant ainsi l'apparition d'une fenêtre de demande d'utilisateur et de mot de passe

Une fois que les champs ont été remplis, l'URL sera de nouveau appelée:

`$_SERVER["PHP_AUTH_User"]` contient le nom du client

`$_SERVER["PHP_AUTH_PW"]` contient le mot de passe

Accès à l'information en PHP

Un exemple de page personnalisée :

```
<?
echo "Bonjour {$_SERVER["PHP_AUTH_USER"]}.";
?>
```

Au lieu d'utiliser le fichier .htaccess, on peut aussi utiliser un formulaire HTML afin de demander le login et le mot de passe à l'usager

La gestion des logins et mots de passe est alors à la charge du programmeur (maximum de tentatives d'essai) :

```
si login et mdp ne sont pas remplis {
   afficher le formulaire
} sinon {
   si login et mdp sont invalides {
      ré-afficher le formulaire
      afficher les erreurs
   } sinon {
      afficher la "vraie" page}}
```


Accès à l'information en PHP

Exemple de script pour forcer l'authentification d'un client qui veut accéder à une page (N. Biot) :

```
<?php
if( !isset($PHP_AUTH_USER) && !isset($PHP_AUTH_PW) ) {
    Header("WWW-Authenticate: Basic realm=\"Authentification
PHPindex\"");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Vous ne pouvez pas accéder à cette page";
    exit;
}
else {
    echo "login : ".$PHP_AUTH_USER."<br>";
    echo "mot de passe : ".$PHP_AUTH_PW.".";
}
?>
```

Au lieu d'afficher les variables globales `$PHP_AUTH_USER` et `$PHP_AUTH_PW`, on peut vérifier la validité du nom d'utilisateur et du mot de passe en envoyant ces informations à une base de données, ou en recherchant dans un annuaire LDAP

Accès à l'information en PHP

La fonction `function auth()` permet la vérification mot de passe et login :

```
$user = "user";
$pwd = "pwd";
function auth(){
    $realm="Authentification PHPindex";
    Header("WWW-Authenticate: Basic realm='".$realm."'");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Vous ne pouvez pas accéder à cette page";
    exit;
}
if( !isset($PHP_AUTH_USER) && !isset($PHP_AUTH_PW) ) {
    auth();
}
else {
    if( $PHP_AUTH_USER==$user && $PHP_AUTH_PW==$pwd ) {
        echo "Bienvenue sur ce site";
    }
    else{
        auth();
    }
}
```


Accès à l'information en PHP

Le but est que le client puisse être reconnu sur chaque page du site

Une fonction appelée à chaque début de script par l'intermédiaire d'un script inclus "auth.inc.php"

Exemple de page :

```
<?
include "auth.inc.php";

# ----------

# Reste du script PHP

# ----------
?>
```

Une fois le user et le mot de passe saisis, les variables sont stockées dans le cache du navigateur. Elles ne sont donc demandées qu'une fois mais testées tout de même à chaque clic

Accès à l'information en PHP

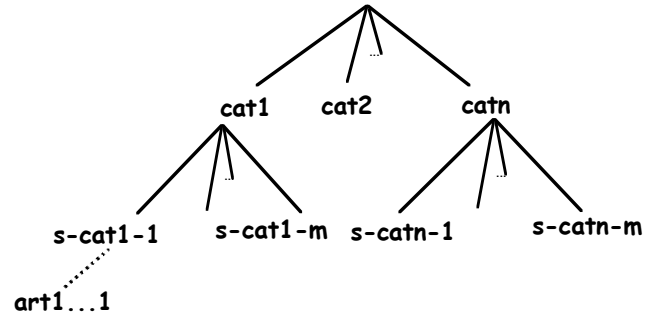
Exemple de fichier `auth.inc.php`

```
<?php
$user = "user";
$pwd = "pwd";
function auth(){
    $realm="Authentification PHPindex";
    Header("WWW-Authenticate: Basic realm='".$realm."'");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Vous ne pouvez pas accéder à cette page";
    // la redirection est impossible, mais page html d'erreur personnalisée
    include "erreur401.html";
    exit;
}
if( !isset($PHP_AUTH_USER) && !isset($PHP_AUTH_PW) ) {
    auth();
}
else {
    if( $PHP_AUTH_USER==$user && $PHP_AUTH_PW==$pwd ) {
        // la suite du script sera exécutée
    }
    else{
        // rappel de la fonction d'identification
        auth();
    }
}
?>
```


Catalogues

Pour rendre navigable une table (d'articles)

- créer une taxinomie de catégories, de sous-catégories, de sous-sous catégories… d'articles en autant de tables secondaires



- Ajouter des clés étrangères aux tables

clé	libellé	valeur	cat	S-cat	…	S-…cat

Catalogues

Exemple de script qui permet d'afficher un tableau de sous-groupe d'articles avec des liens clickables :

```
<?
include("boutique.php")
Init_boutique( );
Catalogue ($Base_R, $Identifiant);
Mysql_connect("$MySQL_Host","$MySQL_User","$MySQL_Passw");
$sgrart = new sousgroupearticle;
$result-mysql("$db","SELECT * FROM sousgroupearticle WHERE NGA
  ='$grart' ");
EditF( "Sous-groupe d'articles : <br><br>" );
echo "<ul>";
$sgrart->getnum($result);
while ($sgrart->z < $sgrart->num)
$sgrart->readrow($result);
  EditF("<li>
<:a href = '$Base_R/article.php?NSCA=$sgrart=>NumSGrArt & NGA=$sgrart-
  >NumGrArt & Identifiant = $Identifiant'>$sgrart->SGrArt </a><li>");
}
echo "</ul>";
?>
```