Un site sécurisé et portable

Vous vous apprêtez à lire un tutoriel rédigé par un membre de ce site. Malgré tout le soin que ce membre a pu apporter au tutoriel, nous ne pouvons pas garantir que les informations contenues sur cette page sont exactes à 100%. Merci de garder cela en tête lorsque vous lirez cette page ;o)

Auteur : pylaterreur
Créé : le 06/01/2008 à 19:24:48
Modifié : Hier à 20:52:21
Noter et commenter ce tutoriel
Imprimer ce tutoriel
Editer ce tutoriel

Dans ce mini-tuto, vous apprendrez à sécuriser votre site de manière automatisée.

Il est recommandé pour cela d'avoir un site ayant une structure similaire à celle donnée dans le tuto d'Asibasth : Includes sécurisés et infinis. Il faut de plus savoir comment éviter les injections SQL.

De plus, vous pourrez rendre votre site "portable", vous n'aurez plus par exemple, besoin de modifier les liens absolus dans votre code lors d'un transfert vers le ftp : fini la galère du remplacer *http://localhost* par *http://monsite.com* ou les changements de mot de passe pour se connecter à MySQL

Sommaire du chapitre :

Sécurité automatisée

Si vous avez bien lu et appliqué le tuto d'Asibasth, vous avez normalement un site basé sur la page root/index.php. Nous allons coder sur cette page-là.

Comme vous le savez, les superglobales $_GET, $_POST et $_COOKIE peuvent contenir tout et nimporte quoi, sachant qu'elles proviennent de l'ordinateur du visiteur. Pour éviter de voir son site hacké, il faut penser à les sécuriser. On leur appliquera un traitement spécial :

- un htmlspecialchars() pour bloquer le html
- un mysql_real_escape_string() sur les variables contenant des chaînes de caractères qui seront utilisées dans une requête SQL
- un intval() sur les variables contenant des nombres qui seront utilisées dans une requête SQL

Il faut maintenant faire comprendre à PHP que telle variable doit subir tel traitement ou tel autre. On va devoir fixer des règles sur les noms des variables envoyées par lien ou par formulaire (ou récupérées à partir d'un cookie).
Les variables numériques destinées à être utilisées dans des requêtes SQL auront un nom comportant le préfixe *int_*.
Les variables contenant des chaînes de caractères qui seront utilisées, elles-aussi, dans des requêtes SQL, auront elles-aussi un préfixe, *string_* cette fois-ci.
Et enfin, toutes les valeurs des superglobales $_GET, $_POST et $_COOKIE seront traitées au htmlspecialchars().

Si vous faîtes **htmlspecialchars($variable);** et que $variable est un array, vous aurez une erreur. La solution consiste à vérifier si la-dite **$variable** est un array au moyen de la fonction is_array(). Si c'est un array, on analyse de la même manière

l'array **$variable** que pour l'array "père".
Une solution simple est la création d'une fonction **securite()**, à qui on enverra nos variables à sécuriser

Voici une solution possible, si vous n'avez pas codé de la même façon, ce n'est pas grave du moment qu'il n'y a pas de trous de sécurité.

Code : PHP - Afficher / masquer les numéros de ligne

```php
<?php
//commence_par() sert à savoir si $variable commence par $debut
function commence_par($variable='',$debut='')
{
	if(substr($variable,0,strlen($debut))==$debut)
		return true;
	else return false;
}

//securite() est le coeur de la sécurisation
function securite($variable='',$type_securite='')
{
	if(is_array($variable))
	{
		foreach($variable as $cle => $element)
		{
			$variable[$cle]=securite($element,$cle);//la fonction fait appel à elle-même, vive la récursivité !
		}
		return $variable;
	}
	else
	{
		if(commence_par($type_securite,'int_')) $variable = intval($variable);
		elseif(commence_par($type_securite,'string_')) $variable = mysql_real_escape_string($variable);
		return htmlspecialchars($variable);
	}
}

//c'est ici qu'il faudra penser à mettre le premier affiche_array($_POST); qui se trouve dans le code suivant

//Le dernier foreach : il fait passer à securite() les trois superglobales $_GET, $_POST et $_COOKIE
foreach(array('GET','POST','COOKIE') as $element)
	${'_'.$element}=securite(${'_'.$element});//ATTENTION : ne pas oublier de récupérer le return de securite(), sinon ce n'est pas sécurisé

?>
```

Pour bien voir l'effet sur un array, nous allons faire un test sur $_POST (pensez bien à écrire le code précédent):

Code : PHP - Afficher / masquer les numéros de ligne

```php
<?php
$_POST = array('<strong>plop</strong>',array('int_membre'=>'<strong>plop</strong>',array('<strong>plop</stro
d'avoir à poster un formulaire pour voir l'effet de securite()

function affiche_array($array='')
{
	echo '<pre>';
	print_r($array);
	echo '</pre>';
}

affiche_array($_POST);//à écrire avant le dernier foreach, sinon vous auriez deux fois un $_POST sécurisé devant les yeux.
$_POST = securite($_POST);
affiche_array($_POST);
?>
```

Cette sécurité est relative, et dépend en fait beaucoup de vous. Vous devez désormais faire très attention aux noms de vos variables, comme récapitulé dans le tableau suivant, sinon le remède serait pire que le mal : vous vous croiriez en sécurité alors que non .

Destination de la variable	Préfixe	Fonction associée
Tout	Aucun préfixe	htmlspecialchars()
MySQL (chaîne de caractère)	**string_**	mysql_real_escape_string()
MySQL (nombre)	**int_**	intval()

Avec tout ça, vous ne devriez plus avoir de problèmes de sécurité avec vos superglobales

PORTABILITÉ ACCRUE

La première sous-partie était un gros morceau, la seconde sera plus relax .
Là-aussi nous travaillerons sur la page root/index.php.

Quand je dis "portabilité", j'entends par là "facilité d'adaptation de son site hébergé en local vers internet, et vice-versa". Vous n'êtes pas sans savoir que les liens absolus internes au site ne sont pas facile à gérer si vous travaillez avec EasyPHP ou Wampserver (ou autre), car il faut modifier les "http://localhost" en "http://monsite.com".

Des liens souples

Dans cette sous-partie, vous apprendrez à faire modifier automatiquement les liens en fonction du nom de domaine. Pour cela, on créera une fonction renvoyant le nom de domaine, en se servant de la superglobale **$_SERVER**.

Pour voir un peu ce que contient $_SERVER, amusez-vous à l'afficher :

Code : PHP - Afficher / masquer les numéros de ligne

```php
<?php
echo '<pre>';
print_r($_SERVER);
echo '</pre>';
?>
```

Un sacré bazar, non ? Essayez de retrouver votre nom de domaine dans cette liste . Je vous laisse chercher avant de continuer...

C'est **$_SERVER['SERVER_NAME']** qui contient votre nom de domaine. Pour avoir l'adresse, il suffit de faire précéder cette variable par "http://". Créons une fonction qui renvoie l'adresse du site, ce retour servira directement pour les liens ou pour le chemin absolu des images.

Code : PHP - Afficher / masquer les numéros de ligne

```php
<?php
function nom_site($partie_de_l_adresse='')
{
        if(empty($partie_de_l_adresse))
                return $_SERVER['SERVER_NAME'];
        elseif($partie_de_l_adresse=='tout')
                return 'http://'.$_SERVER['SERVER_NAME'].'/index.php?page=';//dans le tuto "Includes sécurisés et infinis" d'Asibasth, on inclut le fichier 'root/includes/'.$_GET['page'].'php' (après vérifications, bien entendu ;) )
}
?>
```

Désormais, dans n'importe quelle page PHP, vous pouvez taper :

Code : PHP - Afficher / masquer les numéros de ligne

```php
<?php
echo '<a href="'.nom_site('tout').'forums">Les forums</a>';
?>
```

Le problème de la connexion à MySQL

MySQL demande un mot de passe avant de se connecter, et ce mot de passe diffère suivant que vous êtes en local avec easyPHP ou Wampserver, ou sur votre hébergeur sur le net. Pour cela, on renseignera les mots de passe pour les deux hébergeurs (local et online), et on choisira le mot de passe en fonction du return de nom_site().

Code : PHP - Afficher / masquer les numéros de ligne

```php
<?php
$mysql_array=array(
'localhost'=>array('localhost','root','','nombase'),
'monsite.com'=>array('monsite','root','motdepasse','nombase')
);
mysql_array=$mysql_array[nom_site()];
mysql_connect($mysql_array[0],$mysql_array[1],$mysql_array[2]);
mysql_select_db($mysql_array[3]);
?>
```

Ce code ne devrait pas poser de problème, c'est le plus simple depuis le début (je sais, c'est vache de terminer par le plus facile). Le seul "truc", c'est l'emploi d'arrays imbriqués, mais ce n'est pas la première fois qu'on en trouve dans ce mini-tuto, donc pas de surprise .

Ainsi se conclut cette seconde et dernière sous-partie, dans laquelle on a pu "assouplir les liens" (drôle d'expression, il faut l'avouer) et automatiser la connexion à MySQL selon le serveur hébergeant le site.

Vos superglobales sont désormais sécurisées si vous gardez à l'esprit les règles énoncées dans le tableau (libre à vous de modifier ces règles en modifiant le code PHP).

De plus, vos transferts FTP se déroulent sans avoir à modifier quoi que ce soit dans vos fichiers PHP, vous avez donc un site beaucoup plus souple et facile à mettre à jour, ce qui est essentiel car cela vous laisse plus de disponibilité pour étoffer le contenu de votre site.

Si vous avez le temps et l'envie, faîtes-moi un petit coucou dans les commentaires pour me donner vos impressions et conseils .

Auteur : pylaterreur
Noter et commenter ce tutoriel
Imprimer ce tutoriel
Editer mon tutoriel