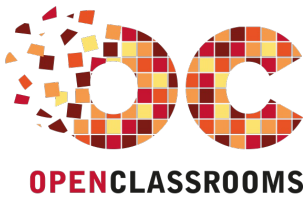
Un anti brute-force léger et rapide

Par ebola



www.openclassrooms.com

Dernière mise à jour le 16/11/2011

Sommaire

Un anti brute-force léger et rapide

Par ebola

Mise à jour : 16/11/2011
Difficulté : Facile

Vous avez sécurisé votre site au maximum, exterminé les failles, protégé vos pages... mais la menace de la force brute pèse sur vous ?

La *force brute* consiste à tester à l'aide d'un script des tas de mots de passe dans vos formulaires, pour accéder à votre administration par exemple (ce qui est plutôt désagréable, je vous l'accorde).

Mettre en place un anti brute-force en utilisant la base de données est la solution la plus souvent envisagée sur les forums du Site du Zér0. Cela fonctionne parfaitement, mais c'est gourmand (une requête pour chaque identification ratée, sans compter les vérifications par la suite).

Dans ce tutoriel, nous allons nous pencher sur une solution plus rapide et qui ne demandera pas l'intervention d'une base de données : un anti brute-force qui fonctionne grâce à des fichiers.
Sommaire du tutoriel :

Est-ce bien utile ?

Mais, comment peut-on contrer une attaque de ce genre ?

C'est très simple : le pirate va essayer plein de mots de passe, mais vraiment beaucoup, car on ne trouve pas un mot de passe aussi facilement.

Le script du pirate va essayer des centaines de mots de passe formés, soit aléatoirement, soit piochés dans une liste qu'il aura téléchargé ou préétabli. On appelle cette seconde option une attaque par dictionnaire.

Mais alors, si c'est aussi compliqué et long pour trouver un mot de passe, à quoi ça sert de s'embêter avec un système de protection ?

C'est vrai, certains développeurs vous diront qu'il est inutile de mettre en place ce genre de protection, car, en théorie, il est très difficile de trouver un mot de passe complexe.

Certes, un pirate mettrait des mois à trouver un mot de passe complexe comme celui-ci : 4gf/ki-2KL@[23]. Mais qu'en est-il des mots de passe de vos membres ?

Il y a peu de chances de voir des membres utiliser ce genre de mot de passe ; certains mettront certainement des mots de passe basiques comme un prénom ou le nom de leur héros préféré. Et là, une attaque par dictionnaire le trouvera en peu de temps. Un

pirate pourra alors s'amuser avec le compte d'un de vos membres et, croyez-moi, c'est sur vous et sur la réputation du site que ça retombera.

Une des solutions consiste simplement à forcer les gens à entrer un mot de passe très complexe lors de leur inscription. Cette solution n'est pas forcement celle à adopter : moins les visiteurs auront d'obstacles lors de leur inscription et de leur navigation, et meilleure sera le souvenir qu'ils garderont de votre site web.

Principe de base

Ici, on ne va pas utiliser de base de données pour notre script (trop lent et trop gourmand pour cette utilisation) mais des fichiers.

Oh, mais tu oublies les sessions et les cookies !

Non, je ne les ai pas oubliés ; c'est en effet une solution. Néanmoins, il y a un petit souci avec eux : il suffit de vider ses cookies pour pouvoir recommencer à essayer des mots de passe. De plus, vous allez voir qu'utiliser des fichiers n'est pas plus difficile !

Le principe est simple : si quelqu'un tente de s'identifier avec un mauvais mot de passe, on crée un fichier nommé *pseudodumembre.tmp* qui contiendra la date du jour et le nombre de tentatives.
Et c'est tout, ce sont les seules informations dont on aura besoin.

À chaque fois qu'une personne tentera de s'identifier, il suffira de regarder si un fichier ayant comme nom son pseudo existe. S'il a atteint le quota de mauvais mots de passe journalier, on bloque l'identification.

À vos claviers, c'est parti

Pour pouvoir appliquer ce tutoriel, vous devez connaître les bases de la gestion de fichiers en PHP.

Commençons par le plus simple : la création du dossier qui contiendra tous les fichiers générés par le script anti brute-force. Vous pouvez créer le dossier où vous le souhaitez et le nommer à votre goût. Le tutoriel sera basé sur un dossier antibrute placé à la racine du site (si vous êtes sous linux, pensez à chmoder le dossier en 777).

C'est fait ? Parfait, on peut continuer !

Voici le script d'identification (*ultra-simplifié*) sur lequel nous allons nous baser pour créer l'anti brute-force. Évidemment celui-ci est une pure invention, vous pouvez appliquer ce tutoriel sur vos propres scripts.

Code : PHP

```
<?php

// Si on tente de s'identifier
if(!empty($_POST['pseudo']) AND !empty($_POST['motdepasse']))
{


$verifications = mysql_query('SELECT pseudo, motdepasse FROM membres
WHERE pseudo = \''.mysql_real_escape_string($_POST['pseudo']).'\'
');

$data_verif = mysql_fetch_assoc($verifications);


    // Si le pseudo existe bien
    if(!empty($data_verif['pseudo']))
    {

        // Si le mot de passe est bon
        if($data_verif['motdepasse'] == trim($_POST['motdepasse']))
        {
            //-----------------------------------------------
```

```
                // Ici Votre script qui identifie le membre
                //-----------------------------------------------
            }
            // Si le mot de passe est faux
            else
            {
                echo 'Mot de passe incorrect';
            }

        }
        // Si le pseudo n'existe pas
        else
        {
            echo 'Pseudonyme incorrect';
        }

    }

    ?>
```

Jusque là, normalement, ça vous est tout à fait familier. Commençons les choses sérieuses.

Avant tout, on va vérifier l'existence d'un fichier indiquant des tentatives d'identification.
Si on trouve un fichier créé dans la journée, on récupère les informations qu'il contient ; on saura alors combien de tentatives il reste au visiteur en question.

On commence par regarder si le fichier existe grâce à la fonction `file_exists()`.

Si la réponse est positive, on ouvre le fichier avec `fopen()` puis on récupère le contenu du fichier dans une variable avec `fgets()`.

Il faut aussi que l'on fasse en sorte que le fichier soit créé s'il n'existe pas encore, ou mis à jour si la date est dépassée. Pour cela, on va donner naissance à une variable `$existence_ft` en cas de retour négatif de la fonction `file_exists()` ou de mauvaise date.

Mais pourquoi ne pas créer / mettre à jour le fichier tout de suite au lieu de créer cette variable ?

Si on créait le fichier tout de suite, il faudrait forcement le faire en se basant sur le pseudonyme entré dans le formulaire par le pirate. On se retrouverait donc avec un fichier par pseudo même s'il n'existe pas ; le pirate pourrait alors faire planter le serveur en essayant des milliers de pseudos bidons ! On créera donc le fichier uniquement si le pseudonyme existe dans la base de données. Comme cela, il y aura un maximum d'un fichier par membre. D'où l'utilité de cette variable dont on vérifiera l'existence au moment venu pour éviter de regarder une deuxième fois si le fichier existe.

Pour ce qui est de la mise à jour de la date, à quoi bon consommer des ressources pour changer la date alors que le membre ne va peut être pas se tromper ? Celui-ci peut très bien s'identifier chaque jour sans problème, ce serait du gaspillage de ressources de s'amuser à changer le contenu du fichier pour rien.

Dans un premier temps, on met `$existence_ft` à 0 si le fichier n'existe pas du tout.

Code : PHP

```
<?php

// Si on tente de s'identifier
if(!empty($_POST['pseudo']) AND !empty($_POST['motdepasse']))
{
```

```
        // On initialise $existence_ft
        $existence_ft = '';


        // Si le fichier existe, on le lit
        if(file_exists('antibrute/'.$_POST['pseudo'].'.tmp'))
        {

        // On ouvre le fichier
        $fichier_tentatives = fopen('antibrute/'.$_POST['pseudo'].'.tmp',
        'r+');

        // On récupère son contenu dans la variable $infos_tentatives
        $contenu_tentatives = fgets($fichier_tentatives);

        }
        // Si le fichier n'existe pas encore, on met la variable
        $existence_ft à 1 et on met les $tentatives à 0
        else
        {
        $existence ft = 1;
        $tentatives = 0;
        }



                $verifications = mysql_query('SELECT pseudo, motdepasse FROM
        membres WHERE pseudo =
        \''.mysql_real_escape_string($_POST['pseudo']).'\' ');

                $data_verif = mysql_fetch_assoc($verifications);

                // Si le pseudo existe bien
                if(!empty($data_verif['pseudo']))
                {

                   // Si le mot de passe est bon
                   if($data_verif['motdepasse'] ==
        trim($_POST['motdepasse']))
                   {
                       //-------------------------------------------------
                       // Ici Votre script qui identifie le membre
                       //-------------------------------------------------
                   }
                   // Si le mot de passe est faux
                   else
                   {
                       echo 'Mot de passe incorrect';
                   }

                }
                // Si le pseudo n'existe pas
                else
                {
                    echo 'Pseudonyme incorrect';
                }


        }

        ?>
```

Si le fichier existe on obtient son contenu dans la variable `$contenu_tentatives`, il ne nous reste plus qu'à traiter ces informations pour pouvoir les utiliser. La fonction `explode()` va nous permettre de couper en morceaux le contenu du fichier selon le séparateur de notre choix. Ici le point virgule (;).

Code : PHP

```php
<?php

// Si on tente de s'identifier
if(!empty($_POST['pseudo']) AND !empty($_POST['motdepasse']))
{


    // On initialise $existence_ft
    $existence_ft = '';


    // Si le fichier existe, on le lit
    if(file_exists('antibrute/'.$_POST['pseudo'].'.tmp'))
    {

        // On ouvre le fichier
        $fichier_tentatives =
fopen('antibrute/'.$_POST['pseudo'].'.tmp', 'r+');

        // On récupère son contenu dans la variable
$infos_tentatives
        $contenu_tentatives = fgets($fichier_tentatives);

// On découpe le contenu du fichier pour récupérer les informations
$infos_tentatives = explode(';', $contenu_tentatives);

    }
    // Si le fichier n'existe pas encore, on met la variable
$existence_ft à 1 et on met les $tentatives à 0
    else
    {
        $existence_ft = 1;
        $tentatives = 0;
    }



        $verifications = mysql_query('SELECT pseudo, motdepasse FROM
membres WHERE pseudo =
\''.mysql_real_escape_string($_POST['pseudo']).'\' ');

        $data_verif = mysql_fetch_assoc($verifications);

        // Si le pseudo existe bien
        if(!empty($data_verif['pseudo']))
        {

           // Si le mot de passe est bon
           if($data_verif['motdepasse'] ==
trim($_POST['motdepasse']))
           {
               //-------------------------------------------------
               // Ici Votre script qui identifie le membre
               //-------------------------------------------------
           }
           // Si le mot de passe est faux
           else
           {
               echo 'Mot de passe incorrect';
           }

        }
        // Si le pseudo n'existe pas
        else
        {
            echo 'Pseudonyme incorrect';
```

```
            }


    }

    ?>
```

Le contenu de notre fichier se présente de cette manière :

01/10/2008;0

La fonction `explode()` a donc créé un tableau contenant tout d'abord la date puis le nombre de tentatives.

- $infos_tentatives[0] contient la date.
- $infos_tentatives[1] contient le nombre de tentatives.

La première chose que l'on va faire, c'est comparer la date du jour à la date que l'on vient de récupérer dans le fichier. Si la date est identique, le fichier est encore valide et on va utiliser le nombre de tentatives qu'il contient. Si au contraire la date ne correspond pas, on met la fameuse variable `$existence_ft` à 1.

On en profite pour créer la variable `$tentatives` que l'on utilisera pour savoir si le quota est dépassé ou non.

Code : PHP

```
<?php

// Si on tente de s'identifier
if(!empty($_POST['pseudo']) AND !empty($_POST['motdepasse']))
{


    // On initialise $existence_ft
    $existence_ft = '';


    // Si le fichier existe, on le lit
    if(file_exists('antibrute/'.$_POST['pseudo'].'.tmp'))
    {

        // On ouvre le fichier
        $fichier_tentatives =
fopen('antibrute/'.$_POST['pseudo'].'.tmp', 'r+');

        // On récupère son contenu dans la variable
$infos_tentatives
        $contenu_tentatives = fgets($fichier_tentatives);

        // On découpe le contenu du fichier pour récupérer les
informations
        $infos_tentatives = explode(';', $contenu_tentatives);


// Si la date du fichier est celle d'aujourd'hui, on récupère le
nombre de tentatives
if($infos_tentatives[0] == date('d/m/Y'))
{
$tentatives = $infos_tentatives[1];
}
// Si la date du fichier est dépassée, on met le nombre de
tentatives à 0 et $existence_ft à 2
```

```
        else
        {
        $existence_ft = 2;
        $tentatives = 0; // On met la variable $tentatives à 0
        }

            }
            // Si le fichier n'existe pas encore, on met la variable
        $existence_ft à 1 et on met les $tentatives à 0
            else
            {
                $existence_ft = 1;
                $tentatives = 0;
            }



                $verifications = mysql_query('SELECT pseudo, motdepasse FROM
        membres WHERE pseudo =
        \''.mysql_real_escape_string($_POST['pseudo']).'\' ');

                $data_verif = mysql_fetch_assoc($verifications);

                // Si le pseudo existe bien
                if(!empty($data_verif['pseudo']))
                {

                   // Si le mot de passe est bon
                   if($data_verif['motdepasse'] ==
        trim($_POST['motdepasse']))
                   {
                       //-------------------------------------------------
                       // Ici Votre script qui identifie le membre
                       //-------------------------------------------------
                   }
                   // Si le mot de passe est faux
                   else
                   {
                       echo 'Mot de passe incorrect';
                   }

                }
                // Si le pseudo n'existe pas
                else
                {
                    echo 'Pseudonyme incorrect';
                }


        }

        ?>
```

Pfiou, ça avance. Maintenant on possède deux variables qui nous fournissent toutes les informations dont on a besoin pour la suite. Il ne reste que deux choses à faire avant que le script soit fonctionnel :

- Couper l'accès à l'identification dès que le quota de tentatives a été atteint.
- Mettre à jour le fichier à chaque fois qu'un mauvais mot de passe est envoyé.

La première tâche est très simple à réaliser, on a déjà le nombre de tentatives dans une variable. Tout ce qu'il reste à faire c'est de le comparer à notre quota.

Code : PHP

```php
<?php

// Si on tente de s'identifier
if(!empty($_POST['pseudo']) AND !empty($_POST['motdepasse']))
{


    // On initialise $existence_ft
    $existence_ft = '';


    // Si le fichier existe, on le lit
    if(file_exists('antibrute/'.$_POST['pseudo'].'.tmp'))
    {

        // On ouvre le fichier
        $fichier_tentatives =
fopen('antibrute/'.$_POST['pseudo'].'.tmp', 'r+');

        // On récupère son contenu dans la variable
$infos_tentatives
        $contenu_tentatives = fgets($fichier_tentatives);

        // On découpe le contenu du fichier pour récupérer les
informations
        $infos_tentatives = explode(';', $contenu_tentatives);


        // Si la date du fichier est celle d'aujourd'hui, on
récupère le nombre de tentatives
        if($infos_tentatives[0] == date('d/m/Y'))
        {
            $tentatives = $infos_tentatives[1];
        }
        // Si la date du fichier est dépassée, on met le nombre de
tentatives à 0 et $existence_ft à 2
        else
        {
            $existence_ft = 2;
            $tentatives = 0; // On met la variable $tentatives à 0
        }


    }
    // Si le fichier n'existe pas encore, on met la variable
$existence_ft à 1 et on met les $tentatives à 0
    else
    {
        $existence_ft = 1;
        $tentatives = 0;
    }


// S'il y a eu moins de 30 identifications ratées dans la journée,
on laisse passer
if($tentatives < 30)
{


        $verifications = mysql_query('SELECT pseudo, motdepasse FROM
membres WHERE pseudo =
\''.mysql_real_escape_string($_POST['pseudo']).'\' ');

        $data_verif = mysql_fetch_assoc($verifications);

        // Si le pseudo existe bien
```

```
            if(!empty($data_verif['pseudo']))
            {

                // Si le mot de passe est bon
                if($data_verif['motdepasse'] ==
    trim($_POST['motdepasse']))
                {
                    //-----------------------------------------------
                    // Ici Votre script qui identifie le membre
                    //-----------------------------------------------
                }
                // Si le mot de passe est faux
                else
                {
                    echo 'Mot de passe incorrect';
                }

            }
            // Si le pseudo n'existe pas
            else
            {
                echo 'Pseudonyme incorrect';
            }


    }
    // S'il y a déjà eu 30 tentatives dans la journée, on affiche un
    message d'erreur
    else
    {
    echo 'Trop de tentatives d\'authentification aujourd\'hui';
    }

    }

    ?>
```

La seconde sera légèrement plus ardue. Si le fichier n'existe pas encore, on le crée et on indique à l'intérieur qu'il y a eu une tentative. Mais s'il a déjà été créé, il va falloir à chaque identification ratée mettre à jour le nombre de tentatives à l'intérieur du fichier. Pour cela on va incrémenter notre variable tentative (+1), placer le curseur juste à l'endroit où se trouve le nombre de tentatives puis écrire le nouveau nombre. Il faudra aussi mettre à jour la date si nécessaire.

Inutile de s'affoler, je suis certain que vous y arriverez très facilement  .

Selon la configuration de votre serveur, l'encodage de vos données, et le type de caractères que vos membres ont le droit d'insérer dans leurs pseudonymes ; il se peut que vous deviez convertir les noms de fichiers pour qu'ils soient créés sans problème.

Code : PHP

```
    <?php

    // Si on tente de s'identifier
    if(!empty($_POST['pseudo']) AND !empty($_POST['motdepasse']))
    {


        // On initialise $existence_ft
        $existence_ft = '';
```

```
    // Si le fichier existe, on le lit
    if(file_exists('antibrute/'.$_POST['pseudo'].'.tmp'))
    {

        // On ouvre le fichier
        $fichier_tentatives =
fopen('antibrute/'.$_POST['pseudo'].'.tmp', 'r+');

        // On récupère son contenu dans la variable
$infos_tentatives
        $contenu_tentatives = fgets($fichier_tentatives);

        // On découpe le contenu du fichier pour récupérer les
informations
        $infos_tentatives = explode(';', $contenu_tentatives);


        // Si la date du fichier est celle d'aujourd'hui, on
récupère le nombre de tentatives
        if($infos_tentatives[0] == date('d/m/Y'))
        {
            $tentatives = $infos_tentatives[1];
        }
        // Si la date du fichier est dépassée, on met le nombre de
tentatives à 0 et $existence_ft à 2
        else
        {
            $existence_ft = 2;
            $tentatives = 0; // On met la variable $tentatives à 0
        }


    }
    // Si le fichier n'existe pas encore, on met la variable
$existence_ft à 1 et on met les $tentatives à 0
    else
    {
        $existence_ft = 1;
        $tentatives = 0;
    }


    // S'il y a eu moins de 30 identifications ratées dans la
journée, on laisse passer
    if($tentatives < 30)
    {


        $verifications = mysql_query('SELECT pseudo, motdepasse FROM
membres WHERE pseudo =
\''.mysql_real_escape_string($_POST['pseudo']).'\' ');

        $data_verif = mysql_fetch_assoc($verifications);

        // Si le pseudo existe bien
        if(!empty($data_verif['pseudo']))
        {

           // Si le mot de passe est bon
           if($data_verif['motdepasse'] ==
trim($_POST['motdepasse']))
           {
               //-----------------------------------------------
               // Ici Votre script qui identifie le membre
               //-----------------------------------------------
           }
           // Si le mot de passe est faux
           else
           {
```

```
        // Si le fichier n'existe pas encore, on le créer
        if($existence_ft == 1)
        {
        $creation fichier = fopen('antibrute/'.$data verif['pseudo'].'.tmp',
        'a+'); // On créer le fichier puis on l'ouvre
        fputs($creation fichier, date('d/m/Y').';1'); // On écrit à
        l'intérieur la date du jour et on met le nombre de tentatives à 1
        fclose($creation_fichier); // On referme
        }
        // Si la date n'est plus a jour
        elseif($existence_ft == 2)
        {
        fseek($fichier_tentatives, 0); // On remet le curseur au début du
        fichier
        fputs($fichier tentatives, date('d/m/Y').';1'); // On met à jour le
        contenu du fichier (date du jour;1 tentatives)
        }
        else
        {
        fseek($fichier tentatives, 11); // On place le curseur juste devant
        le nombre de tentatives
        fputs($fichier tentatives, $tentatives + 1); // On ajoute 1 au
        nombre de tentatives
        }


                    echo 'Mot de passe incorrect';
                    }

                }
                // Si le pseudo n'existe pas
                else
                {
                    echo 'Pseudonyme incorrect';
                }


            }
            // S'il y a déjà eu 30 tentatives dans la journée, on affiche
        un message d'erreur
            else
            {
                echo 'Trop de tentatives d\'authentification aujourd\'hui';
            }



        // Si on a ouvert un fichier, on le referme (eh oui, il ne faut pas
        l'oublier)
        if($existence_ft != 1)
        {
        fclose($fichier_tentatives);
        }

        }

        ?>
```

Quelques explications s'imposent. Lorsque le fichier n'existe pas encore, c'est très simple : on le crée. Par contre, si la date n'est plus à jour, on va réécrire par dessus. Au tout début du script, on a ouvert et lu le fichier ; celui-ci est donc toujours ouvert. Tout ce qu'il nous reste à faire c'est de replacer le curseur au début avec `fseek($fichier_tentatives, 0);` puis d'écrire à la place du contenu actuel la date du jour suivie du nouveau nombre de tentatives (qui est logiquement 1 ici) via `fputs($fichier_tentatives, date('d/m/Y').';1');`. L'espace derrière le chiffre 1 n'est pas une erreur, il faut absolument le laisser. Si le fichier contenait par exemple 10/02/2008;25 (25 tentatives le 10/02/2008) et que vous écrivez par dessus 10/10/2008;1 vous obtiendriez 10/10/2008;15.
Le nouveau contenu étant 1 caractère plus petit que le précédent, le dernier chiffre est resté et fausse complètement le nombre de tentatives. D'où l'utilité d'écrire un espace à la fin du nouveau contenu pour écraser un éventuel caractère gênant.

Si le fichier existe et est à la bonne date, il n'y a qu'à incrémenter le nombre de tentatives. Inutile de s'amuser à réécrire tout le fichier. On place le curseur juste devant le nombre de tentatives grâce à `fseek($fichier_tentatives, 11);`, puis on écrit par dessus le nouveau nombre. Pourquoi utiliser 11 avec la variable fseek ? tout simplement parce que la date suivit du point virgule fera toujours 11 caractères.

Voilà ! Le script anti brute-force est maintenant fonctionnel.
Celui-ci n'autorisera que 30 tentatives par jour ; cela découragera pas mal de pirates. Si vous estimez que ce n'est pas suffisant, il est très simple de baisser le nombre de tentatives.

Bien que le système soit fonctionnel, vous pouvez si vous le souhaitez lire le chapitre suivant pour l'améliorer. Si vous estimez avoir appris bien assez de choses pour aujourd'hui, vous pouvez le lire plus tard ou laisser votre script tel quel.

Ajout d'un systéme de notification

Maintenant que votre anti brute-force fonctionne, il sera beaucoup plus long pour un pirate de réussir à récupérer le mot de passe d'un de vos comptes. Néanmoins il reste deux petits problèmes :

- Un pirate particulièrement acharné pourrait quand même, au bout de quelques temps, réussir à trouver un mot de passe.
- Si un pirate continue malgré la protection à tenter de récupérer un mot de passe, il pourrait empêcher la personne réelle de s'identifier.

Il n'y a malheureusement techniquement rien d'autre à faire pour empêcher quelqu'un d'essayer des mots de passe, ne serai-ce que 30 par jour. On va donc faire en sorte d'être informé à chaque fois qu'un quota est dépassé ; étant donné que les personnes qui se trompent 30 fois de mot de passe sur la journée sont plutôt rares, ces notifications seront généralement légitimes.

La mise en place de ce système est très simple, il suffit d'envoyer un mail à un administrateur grâce à la fonction mail() lorsque un compte est arrivé à 30 tentatives.

Code : PHP

```php
<?php

// Si on tente de s'identifier
if(!empty($_POST['pseudo']) AND !empty($_POST['motdepasse']))
{


    // On initialise $existence_ft
    $existence_ft = '';


    // Si le fichier existe, on le lit
    if(file_exists('antibrute/'.$_POST['pseudo'].'.tmp'))
    {

        // On ouvre le fichier
        $fichier_tentatives =
fopen('antibrute/'.$_POST['pseudo'].'.tmp', 'r+');

        // On récupère son contenu dans la variable
$infos_tentatives
        $contenu_tentatives = fgets($fichier_tentatives);

        // On découpe le contenu du fichier pour récupérer les
informations
        $infos_tentatives = explode(';', $contenu_tentatives);


        // Si la date du fichier est celle d'aujourd'hui, on
récupère le nombre de tentatives
        if($infos_tentatives[0] == date('d/m/Y'))
        {
```

```
            {
                $tentatives = $infos_tentatives[1];
            }
            // Si la date du fichier est dépassée, on met le nombre de
tentatives à 0 et $existence_ft à 2
            else
            {
                $existence_ft = 2;
                $tentatives = 0; // On met la variable $tentatives à 0
            }


        }
        // Si le fichier n'existe pas encore, on met la variable
$existence_ft à 1 et on met les $tentatives à 0
        else
        {
            $existence_ft = 1;
            $tentatives = 0;
        }


        // S'il y a eu moins de 30 identifications ratées dans la
journée, on laisse passer
        if($tentatives < 30)
        {


            $verifications = mysql_query('SELECT pseudo, motdepasse FROM
membres WHERE pseudo =
\''.mysql_real_escape_string($_POST['pseudo']).'\' ');

            $data_verif = mysql_fetch_assoc($verifications);

            // Si le pseudo existe bien
            if(!empty($data_verif['pseudo']))
            {

                // Si le mot de passe est bon
                if($data_verif['motdepasse'] ==
trim($_POST['motdepasse']))
                {
                    //-------------------------------------------------
                    // Ici Votre script qui identifie le membre
                    //-------------------------------------------------
                }
                // Si le mot de passe est faux
                else
                {

                    // Si le fichier n'existe pas encore, on le créé
                    if($existence_ft == 1)
                    {
                        $creation_fichier =
fopen('antibrute/'.$data_verif['pseudo'].'.tmp', 'a+'); // On créé
le fichier puis on l'ouvre
                        fputs($creation_fichier, date('d/m/Y').';1'); //
On écrit à l'intérieur la date du jour et on met le nombre de
tentatives à 1
                        fclose($creation_fichier); // On referme
                    }
                    // Si la date n'est plus a jour
                    elseif($existence_ft == 2)
                    {
                        fseek($fichier_tentatives, 0); // On remet le
curseur au début du fichier
                        fputs($fichier_tentatives, date('d/m/Y').';1 ');
// On met à jour le contenu du fichier (date du jour;1 tentatives)
                    }
                    else
```

```
                {

        // Si la variable $tentatives est sur le point de passer à 30, on
        en informe l'administrateur du site
        if($tentatives == 29)
        {
        $email_administrateur = 'Email de administrateur du site';

        $sujet_notification = '[Site] Un compte membre a atteint son quota';

        $message notification = 'Un des comptes a atteint le quota de
        mauvais mots de passe journalier :';
        $message notification .= $data_verif['pseudo'].' -
        '.$ SERVER['REMOTE ADDR'].' -
        '.gethostbyaddr($_SERVER['REMOTE_ADDR']);

        mail($email administrateur, $sujet_notification,
        $message_notification);
        }

                        fseek($fichier_tentatives, 11); // On place le
        curseur juste devant le nombre de tentatives
                        fputs($fichier_tentatives, $tentatives + 1); //
        On ajoute 1 au nombre de tentatives
                    }


                echo 'Mot de passe incorrect';
                }

            }
            // Si le pseudo n'existe pas
            else
            {
                echo 'Pseudonyme incorrect';
            }


        }
        // S'il y a déjà eu 30 tentatives dans la journée, on affiche
        un message d'erreur
        else
        {
            echo 'Trop de tentatives d\'authentification aujourd\'hui';
        }



        // Si on a ouvert un fichier, on le referme (eh oui, il ne faut
        pas l'oublier)
        if($existence_ft != 1)
        {
        fclose($fichier_tentatives);
        }

    }

    ?>
```

Et voilà, l'administrateur du site sera informé à chaque fois que le quota d'un compte sera atteint ! Il recevra :

- Le pseudonyme du compte en question.
- L'adresse IP du pirate présumé.
- Des détails tirés de l'adresse IP permettant généralement de déduire le Fournisseur d'Accès à Internet du pirate présumé.

Évidemment, chaque mail ne voudra pas forcement dire qu'un pirate attaque votre site web. Les membres qui se trompent de mot de passe 30 fois sur une journée existent, ça peut arriver à tout le monde de s'énerver sur un trou de mémoire et essayer plein de mots de passe. Vous pourrez commencer à vous inquiéter si vous recevez un mail pour le même compte plusieurs jours de suite. À vous alors de réagir en conséquence : bloquer le compte attaqué, contacter son propriétaire et vous tournez si nécessaire vers la justice de votre pays pour tenter de vous débarrasser du malotru responsable des attaques.

Si votre site connait un fort trafic, vous pouvez créer une boîte de messagerie rien que pour recevoir les notifications. Vous pourriez aussi choisir d'envoyer les notifications par message privé ou de les insérer dans la base de données. Et bien plus encore.

Ce tutoriel touche à sa fin, mais il est néanmoins amené à évoluer au fil des remarques et commentaires des zéros. Passez jeter un coup d'œil sur les évolutions de temps en temps.

Q.C.M.

Le premier QCM de ce cours vous est offert en libre accès.
Pour accéder aux suivants

Connectez-vous Inscrivez-vous

Qu'est ce qu'une attaque par force brute ?

- ◯ Une faille de securité dans un site web.
- ◯ Une technique qui permet de trouver les mots de passe en essayant des centaines de possibilités.
- ◯ Une technique qui consiste à detruire un site web en cliquant dessus à toute vitesse.

Quelle technique faut-il employer pour se prémunir de ces attaques ?

- ◯ On utilise la superglobale $_SERVER['adresse_du_pirate'] pour obtenir son adresse et lui mettre une bonne claque histoire de lui faire passer l'envie de recommencer.
- ◯ On ne peut rien faire malheureusement.
- ◯ On limite le nombre de tentatives d'identification possibles.

Correction !

Statistiques de réponses au QCM

Vous êtes maintenant à l'abri des attaques par brute-force.

Encore un souci de moins, ça fait plaisir non ?

Partager

Ce tutoriel a été corrigé par les zCorrecteurs.