I.U.T.
Université d'Artois

Sécurité des serveurs web

Corrigés des TPs de configuration d'Apache et de PHP

1 Configuration d'Apache (1)

M1. *Quels sont les exécuteurs des processus du serveur* ***httpd*** *?* Il s'agit de `root` et `apache`.

```
[parrain@mazurka licpro]$ ps  u -C httpd2
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root      3789  0.0  1.8 12060 4832 ?        S    17:22   0:00 httpd2 -f ...
apache    3798  0.0  1.9 12272 4940 ?        S    17:22   0:00 httpd2 -f ...
apache    3799  0.0  1.9 12272 4936 ?        S    17:22   0:00 httpd2 -f ...
apache    3800  0.0  1.9 12272 4936 ?        S    17:22   0:00 httpd2 -f ...
apache    3801  0.0  1.9 12272 4936 ?        S    17:22   0:00 httpd2 -f ...
apache    3802  0.0  1.9 12272 4936 ?        S    17:22   0:00 httpd2 -f ...
```

Inspectez la directive ***User*** *donnée dans* ***httpd.conf****.*

```
User apache
Group apache
```

M4. *Repérez le numéro du port écouté par le serveur en notant le paramètre de la commande Listen se trouvant dans httpd.config.*

```
<IfDefine APACHEPROXIED>
    Listen 8080
</IfDefine>
<IfDefine !APACHEPROXIED>
    Listen 80
</IfDefine>
```

M6. *Changez le répertoire racine des documents publiés de la manière suivante. Créez le répertoire* ***<homeUserDir>****....*

```
DocumentRoot /home/parrain/web
# DirectoryIndex: Name of the file or files to use as a pre-written HTML
# directory index.  Separate multiple entries with spaces.
#
DirectoryIndex principal.html
# This should be changed to whatever you set DocumentRoot to.
#
<Directory /home/parrain/web>
# This may also be "None", "All", or any combination of "Indexes",
# "Includes", "FollowSymLinks", "SymLinksifOwnerMatch", "ExecCGI", or "MultiViews".
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
    Options -Indexes FollowSymLinks MultiViews
# Donne la possibilité de définir des .htaccess localement dans les
# répertoires. cf. TP Apache 3.
    AllowOverride All
</Directory>
```

M11. *Rajoutez au fichier* ***httpd.conf*** *une directive permettant de rajouter l'option* ***Indexes*** *au répertoire* ***<homeUserDir>/web/infos2****.*

```
<Directory /home/parrain/web/infos2>
    Options Indexes
</Directory>
```

M15. *Modifiez httpd.conf pour faire en sorte que s'affiche un texte particulier pour les erreurs 403 et 404 qui surviendrait dans la recherche d'un document dans /web/info1.*

```
<Directory "/home/parrain/web/infos1">
     ErrorDocument 403 "Désolé mais vous n'avez pas accès au document demandé ..."
     ErrorDocument 404 "Désolé mais le document demandé est introuvable ..."
</Directory>
```

M16. *Créez le répertoire <userHomeDir>/web/erreurs. Créez dans ce répertoire les fichiers erreur403.html et erreur404.html contenant respectivement une page pour l'erreur 403 et une page pour l'erreur 404....*

```
Alias /error/ "/home/parrain/web/erreurs"
<Directory "/home/parrain/web/erreurs">
       AllowOverride None
       Options IncludesNoExec
       AddOutputFilter Includes html
       AddHandler type-map var
       Order allow,deny
       Allow from all
</Directory>
ErrorDocument 403 /error/erreur403.html
ErrorDocument 404 /error/erreur404.html
```

M20. *Repérez la directive ScriptAlias dans le fichier httpd.conf. Commentez là. Rajoutez une ligne indiquant que le répertoire contenant les scripts est <homeUserDir>/web/programmes...*

```
ScriptAlias /cgi-bin/ /home/parrain/web/programmes/
<Directory /var/www/cgi-bin>
    AllowOverride All
    Options ExecCGI
</Directory>
```

et le fichier `bonjour.sh`

```
#!/bin/sh
echo "content-type: text/html"
echo
echo "<HTML><HEAD><TITLE>Le shell bonjour</TITLE></HEAD>"
echo "<BODY><H1><B>Bonjour !</B></H1></BODY></HTML>"
```

M21. *Créez un programme similaire au précédent mais réalisé en langage C et qui portera le nom bonjourC. Il sera également placé dans le sous répertoire programmes.*

```
#include <stdio.h>
int main(){
  printf("content-type: text/html\n");
  printf("\n");
  printf("<HTML><HEAD><TITLE>bonjourC</TITLE></HEAD>");
  printf("<BODY><H1><B>Bonjour !</B></H1></BODY></HTML>");
  }
```

M22. *Réalisez un programme en shell appelé afficheEnv permettant d'afficher les variables d'environnement lors de l'appel du script. Ce programme est toujours placé dans le sous-répertoire programmes.*

```
#!/bin/sh
echo "content-type: text/plain"
echo
echo "Le script afficheEnv"
env
```

M24. *Créez le sous-répertoire <homeUserDir>/web/php. Créez dans ce répertoire un fichier appelé bonjour.php qui permet d'afficher bonjour! en gras à l'aide d'au moins une instruction php. Testez ce programme en ouvrant l'url http ://localhost/php/bonjour.php.*

```
<HTML><HEAD>
<TITLE>bonjour.php</TITLE>
</HEAD><BODY>
<?PHP echo "<B>Bonjour !</B>" ?>
</BODY></HTML>
```

2 Configuration d'Apache (2)

M0. *Créez un script shell appelé afficheParametres retournant une page Web (un document HTML) affichant les paramètres associées à l'URL (ce qu'il y a après le ? dans l'URL). La variable d'environnement contenant ces paramètres est la variable* `QUERY_STRING`*. Ce script sera placé dans le répertoire* `<homeUserDir>/web/programmes`*. Testez-le.*

```
#!/bin/sh

cat <<EOF
Content-Type: text/html

EOF

cat <<EOF
<html><head></head><body>
$QUERY_STRING
</body></html>
EOF
```

M1. *Essayez d'injecter du code HTML à travers le script précédent...*

Avec firefox, ou mozilla, par exemple, la requête
`http ://localhost/cgi-bin/afficheParametres.sh ?<b>bonjour</b>` est transformée en
`http ://localhost/cgi-bin/afficheParametres.sh ?%3Cb%3Ebonjour%3C/b%3E`. Les caractères spéciaux sont échappés pour éviter les attaques de cross-scripting. Si on utilise `lwp-request`, même comportement. Mais si vous utilisez `Konqueror` (navigateur de KDE), ou `curl` un utilitaire de requête HTTP en ligne, la commande passe en clair...Même chose pour
`http ://localhost/cgi-bin/afficheParametres.sh ?<script>window.alert('Bonjour !')</script>`
qui provoque l'ouverture d'une fenêtre avec `Konqueror`. Ce sont les clients qui sont à l'origine de l'échappement des caractères spéciaux.

M2. *Mêmes questions avec un script PHP.*

```
<html><body><ul>
<?
echo '<li>SERVER_NAME '.$_SERVER['SERVER_NAME'];
echo '<li>HTTP_HOST '.$_SERVER['HTTP_HOST'];
echo '<li>HTTP_USER_AGENT '.$_SERVER['HTTP_USER_AGENT'];
echo '<li>QUERY_STRING '.$_SERVER['QUERY_STRING'];
?>
</ul></body></html>
```

On retrouve le même comportement : sécurisé avec certains navigateurs (`firefox`, ...) ou permissif avec d'autres `Konqueror`.

M3. *Créez dans le répertoire* `<homeUserDir>/web/programmes` *un script shell appelé* `afficheAgent` *créant une page Web appelée* `infoClientNavigateur.html` *dans le sous-répertoire* `<homeUserDir>/web/infos2`*....*

```
#!/bin/sh
cat <<EOF
Content-Type: text/html

EOF
echo "<HTML><TITLE>AfficheAgent</TITLE>"
echo "<BODY>Ouvrez l'url "
echo "<A HREF="http://localhost/infos2/infoClientNavigateur.html">"
echo " http://localhost/infos2/infoClientNavigateur.html</A>"
echo "</BODY></HTML>"

cat <<EOF >/home/parrain/web/infos2/infoClientNavigateur.html
<html>
<body>
<ul>
<li>SERVER_NAME : $SERVER_NAME
<li>HTTP_HOST : $HTTP_HOST
<li>HTTP_USER_AGENT : $HTTP_USER_AGENT
```

```
</ul>
</body>
</html>
EOF
```

On exécute ensuite les requêtes suivantes (les retours à la ligne dans la commande sont là pour la mise en forme de ce corrigé) :

```
[parrain@mazurka web]$ lwp-request
-H "User-Agent: <b>coucou</b>"
http://localhost/cgi-bin/affichеAgent.sh

<HTML><TITLE>AffichеAgent</TITLE>
<BODY>Ouvrez l'url
<A HREF="http://localhost/infos2/infoClientNavigateur.html">
http://localhost/infos2/infoClientNavigateur.html</A>
</BODY></HTML>
[parrain@mazurka web]$ lwp-request
http://localhost/infos2/infoClientNavigateur.html
<html>
<body>
<ul>
<li>SERVER_NAME : mazurka.hd.free.fr
<li>HTTP_HOST : localhost
<li>HTTP_USER_AGENT : <b>coucou</b>
</ul>
</body>
</html>
```

AïeAïe ! Et pire encore, on aurait pu injecter un script. Ici, l'exmple est gentil (**M4.** et **M5.**) :
`insereHeure.sh` :

```
#!/bin/sh
cat <<EOF
Content-Type: text/html

EOF
date +%T >/home/parrain/web/infos2/heures
```

et maintenant la requête :

```
[parrain@mazurka php]$ lwp-request
-H "User-Agent: <img src="http://localhost/cgi-bin/insereHeure.sh"/>"
http://localhost/cgi-bin/affichеAgent.sh

<BODY>Ouvrez l'url
<A HREF="http://localhost/infos2/infoClientNavigateur.html">
http://localhost/infos2/infoClientNavigateur.html</A>
</BODY></HTML>
[parrain@mazurka php]$ lwp-request
http://localhost/infos2/infoClientNavigateur.html
<html>
<body>
<ul>
<li>SERVER_NAME : mazurka.hd.free.fr
<li>HTTP_HOST : localhost
<li>HTTP_USER_AGENT : <img src="http://localhost/cgi-bin/insereHeure.sh"/>
</ul>
</body>
</html>
```

L'utilisateur croit charger une image, et il exécute un script.

M6. *...Créez une page HTML `formulaire.html` contenant un formulaire avec un champ de saisie et créez `traitement.php`...*
`formulaire.html`

```
<html>
<body>
Ceci est un formulaire. Votre message :
<form action="traitement.php" method="post">
<textarea name="champ" rows="1" cols="20"></textarea>
<input type="submit" value="C'est parti!">
</form>
</body>
</html>
```

`traitement.php`

```
<html><body>
Votre message :
<b><? echo $_POST["champ"]; ?></b>
</body></html>
```

Si l'option `Magic_quotes_gpc` pour la configuration de PHP (dans le fichier `php.ini`) est à `Off`, alors l'attaque fonctionne bien (on peut injecter du code HTML dans `champ`). Si par contre cette option est à `On` (c'est l'option *Magic quotes for incoming GET/POST/Cookie data*), l'attaque échoue. Pour regarder ce qui se passe, transformons le formulaire pour qu'il envoie ses données avec la méthode `GET`. Voilà la donnée transmise quand on demande l'exécution d'un script dans la zone de saisie :

```
champ=%3Cscript%3Ewindow.alert%28%27Bonjour!%27%29%3C%2Fscript%3E
```

Dans ce cas, on peut configurer sur le serveur l'échappement des caractères spéciaux. Vous noterez dans `php.ini` la présence d'une variable `magic_quotes_runtime` (*Magic quotes for runtime-generated data, e.g. data from SQL, from exec(), etc.*).

M7. *Créez une page html affichant l'heure et le contenu du répertoire avec des commandes SSI.*
Dans `httpd.conf`

```
AddOutputFilter INCLUDES .shtml
<Directory /home/parrain/web/ssi>
    Options +Includes
</Directory>
<Directory /home/parrain/web/infos2>
    Options Indexes
</Directory>
```

Fichier `ssi1.shtml`

```
<html><body>
<!--#echo var="DATE_LOCAL" -->
<!--#exec cmd="ls ../infos2" -->
</body></html>
```

M8. *Créez un shell `afficheAgentSSI.sh`...*

```
#!/bin/sh
cat <<EOF
Content-Type: text/html

EOF
cat <<EOF >/home/parrain/web/ssi/infoClientNavigateur.shtml
<html><body><ul>
<li>SERVER_NAME : $SERVER_NAME
<li>HTTP_HOST : $HTTP_HOST
<li>HTTP_USER_AGENT : $HTTP_USER_AGENT
</ul></body></html>
EOF
```

M9. *Et la requête...*

```
lwp-request -H "User-Agent: <!--#exec cmd="ls" -->"
http://localhost/cgi-bin/afficheAgentSSI.sh
```

3 Configuration d'Apache (3)

M0. *Créez un répertoire <homeUserDir>/web/confidentiel1 ...*
Dans `httpd.conf`

```
<Directory /home/parrain/web/confidentiel1/>
    order deny,allow
    deny from all
    allow from 127.0.0.1
    options indexes
</Directory>
```

M2. *Créez le répertoire confidentiel3 ...*
Dans `httpd.conf`

```
<Directory /home/parrain/web/confidentiel3/>
    AuthType Basic
    AuthName secret
    AuthUserFile /home/parrain/web/confidentiel3/.htpasswd
    Require valid-user
</Directory>
```

4 Configuration de PHP

Q3. *Quelles sont les variables à positionner pour effectuer cela ?*

```
display_errors = Off
log_errors = On
error_log = "error.log"
```

Q4. Q5. Q6. Q7. *Les paramètres de sécurité*
Dans `php.ini`

```
register_globals = On
```

Question5.php :

```
<?
session_start();
if ($ok) echo system("ls");
else echo "bouh";
?>
```

Question7.php :

```
<?
session_start();
$ok=false;
?>
<html><body><a href="q5.php">Clickez ici!</a>
</body></html>
```

Dans PHP, les variables ne se déclarent pas (elles sont déclarées à la première utilisation), et il n'y a pas non plus d'obligation de les initialiser. De plus, les informations passées en argument des requêtes http par les méthodes `GET` ou `POST` sont accessibles directement par leur nom, comme une quelconque variable PHP. Cette particularité du langage est un gros trou de sécurité, comme vous avez pu le voir (la requête
`http ://localhost/php/question5.php?ok=true`
provoque l'affichage du contenu du répertoire, ce qui ne devait pas être voulu à l'origine par le programmeur).

Depuis la version 4 de PHP, la variable `register_globals`, si elle est positionnée à `On`, impose l'accès aux variables globales par les tableaux `$_SESSION[$nom_var], $_GET[$nom_var]` ou `$_POST[$nom_var]` suivant leur origine. Le code ainsi réécrit de `question5.php` ne serait plus sensible à la même attaque :

```
<?
session_start();
if ($_SESSION["ok"]) echo system("ls");
else echo "bouh";
?>
```