I.U.T.
Université d'Artois

Sécurité des serveurs web

Fiche de TP n.3 – Apache

1 Injection de code exécuté du client

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.

M1. Essayez d'injecter du code HTML à travers le script précédent, i.e. à travers les paramètres de l'URL `http ://localhost/cgi-bin/afficheParametres` : essayez par exemple de faire retourner par le serveur une page Web affichant *Bonjour !* en gras, essayez également de faire retourner une page ouvrant en Javascript une fenêtre avec le message *Bonjour !* (instruction window.alert("Bonjour !")). Est-ce que cela fonctionne ?

M2. Reprenez les manipulations **M0** et **M1** mais en utilisant une page Web contenant du code PHP. Cette page se trouvera dans le fichier appellé `afficheParametres.php` placé dans le répertoire `<homeUserDir>/web/php`. La variable globale php `$_SERVER` est un tableau contenant des informations sur le serveur et la requête. En particulier

- `$_SERVER['SERVER_NAME']` est le nom du serveur exécutant le script,
- `$_SERVER['HTTP_HOST']` est la valeur de l'en-tête `Host :` si elle existe ;
- `$_SERVER['HTTP_USER_AGENT']` est le contenu de l'en-tête `User_Agent :` si elle existe (une chaîne décrivant le client HTTP effectuant la requête) ;
- `$_SERVER['QUERY_STRING']` correspond à la chaîne des paramètres de la requête.

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`. Cette page contiendra les informations sur le navigateur du client (la valeur de l'en-tête `User_Agent :`, i.e. la variable d'environnement `HTTP_USER_AGENT`) de la dernière personne ayant visitée l'URL
`http ://localhost/cgi-bin/afficheAgent`
De nouveau injectez le code HTML précédent, mais à travers ce nouveau script maintenant, vous utiliserez pour cela l'utilitaire `lwp-request`. Est-ce que cela fonctionne ?

M4. Créez dans le répertoire `<homeUserDir>/web/programmes` un script shell appelé `insereHeure` permettant de juste rajouter l'heure courante dans un fichier appelé heures se trouvant dans `<homeUserDir>/web/infos2`
Testez ce script.

M5. Injectez du code HTML au script `afficheAgent` de manière à ce que les clients récupérant la page `infoClientNavigateur.html` lancent à leur insu le script `insereHeure` (injectez du code HTML utilisant la balise `IMG` par exemple).

M6. Dans le répertoire `<homeUserDir>/web/php`, créez une page HTML `formulaire.html` contenant un formulaire avec un champ de saisie et créez `traitement.php` permettant de traiter le formulaire en affichant la donnée saisie en gras (balise `<strong>`). La valeur de la saisie se récupère par `$_GET['nomSaisie']` (pour le méthode `GET`), `$_POST['nomSaisie']` (pour le mode `POST`). Testez ces fichiers. Injectez à travers le formulaire le code `<emph> Bonjour ! </emph>` puis le code, `<SCRIPT> window.alert('Bonjour !') </SCRIPT>`. Que se passe t-il ? Mettez à `Off` la variable `magic_quotes_gpc` dans le fichier `/etc/php.ini` et recommencez l'injection du code.

2 Injection de code exécuté par le serveur

Les directives SSI (Server Side Includes) sont des directives incluses dans des pages HTML qui sont évaluées par le serveur au moment où les pages sont délivrées. La syntaxe d'une directive SSI est de la forme :

```
<!--#element attribut1=valeur1 attribut2=valeur2 ... ->
```

où `element` est le nom de la commande SSI et `attribut1`, `attribut2`, ... sont des paramètres auxquels sont associées les valeurs `valeur1`, `valeur2`, ... Voici quelques exemples :

- `<!--#echo var=''DATE_LOCAL'' encoding=''url'' -->` pour afficher la variable SSI `DATE_LOCAL` (la date locale) en encodant la chaîne avec un type url (on remplace les caractères spéciaux avec un code commençant avec %). Si on ne veut aucun encodage, on spécifie `none`. Il existe également les variables `DOCUMENT_NAME` (le nom du document demandé), `DATE_GMT` (la date GMT), `DOCUMENT_URI` (l'URL du document demandé par le client) ;
- `<!--exec cgi=''/cgi-bin/script' -->` pour exécuter le script CGI script ;
- `<!--exec cmd=''ls'' -->` pour exécuter la commande ls.

Il existe d'autres commandes SSI.

Pour permettre au serveur d'évaluer les directives SSI des fichiers d'un répertoire il faut que l'option `Includes` soit mise. L'option `IncludesNOEXEC` permet également l'évaluation des directives SSI sauf celles contenant la commande `exec`. Dans le fichier `httpd.conf`, la directive `AddOutputFilter` permet d'indiquer les extensions des fichiers qui vont contenir éventuellement des directives SSI et qui vont être filtrés pour évaluation. Par exemple, pour filtrer les fichiers d'extension `.shtml` on écrirra :

```
AddOutputFilter INCLUDES .shtml
```

M7. Vérifiez dans le fichier `httpd.conf` que les fichiers `.shtml` sont bien filtrés pour l'évaluation des directives SSI. Définissez le répertoire `<homeUserDir>/web/ssi`. Rajoutez dans `httpd.conf` une directive permettant l'évaluation des directives SSI des fichiers de ce répertoire. Créez dans ce répertoire une page HTML ayant pour nom `ssi1.shtml` affichant l'heure et la liste des fichiers `<homeUserDir>/web/infos2` à l'aide de directives SSI. Testez cette page.

M8. Créez dans le répertoire `<homeUserDir>/web/programmes` un script shell appelé `afficheAgentSSI` créant une page Web appelée `infoClientNavigateur.shtml` dans le sous-répertoire `<homeUserDir>/web/ssi`

Cette page contiendra les informations sur le navigateur du client (la valeur de l'en-tête `User_Agent :`, i.e. la variable d'environnement `HTTP_USER_AGENT`) de la dernière personne ayant visitée l'URL `http ://localhost/cgi-bin/afficheAgentSSI`

M9. Injectez des directives SSI dans `<homeUserDir>/web/ssi/infoClientNavigateur.shtml` à l'aide du script `afficheAgentSSI`.