Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
PREMIER MINISTRE

S.G.D.S.N
Agence nationale de la sécurité des systèmes d'information
CERTA

Paris, le 20 mars 2007
N° CERTA-2007-INF-002

Affaire suivie par :
CERTA

NOTE D'INFORMATION DU CERTA

Objet : Du bon usage de PHP

Conditions d'utilisation de ce document : http://www.certa.ssi.gouv.fr/certa/apropos.html
Dernière version de ce document : http://www.certa.ssi.gouv.fr/site/CERTA-2007-INF-002

Gestion du document

Référence	CERTA-2007-INF-002
Titre	Du bon usage de PHP
Date de la première version	20 mars 2007
Date de la dernière version	–
Source(s)	
Pièce(s) jointe(s)	Aucune

TAB. 1 – *Gestion du document*

Une gestion de version détaillée se trouve à la fin de ce document.

1 Introduction

Parmi les incidents traités par le CERTA, il y a beaucoup d'intrusions dans des sites WEB qui s'appuyent sur des applications développées avec le langage PHP. L'objet de cette note d'information est d'attirer l'attention sur les risques spécifiques liés à ce langage et sur les moyens de limiter l'impact des attaques contre ce type d'application.

2 PHP est-il plus dangereux que d'autres langages ?

Aucun langage de programmation ne garantit la sécurité des applications avec lequel elles sont codées. Toutefois il existe plusieurs facteurs de risques spécifiquement liés à PHP.

– PHP offre une grande facilité pour développer des sites dynamiques (c'est à dire un site qui permet l'interaction avec une base de données ou avec le visiteur : forum, journal WEB, Wiki, agenda, webmail, ...).
  Un site dynamique est fragile et plus difficile à sécuriser qu'un site statique, parce qu'on ne peut pas prévoir le comportement des visiteurs. Un utilisateur malintentioné va essayer, lors de sa consultation du site, de soumettre des données astucieusement construites dans le but de contourner des protections.
  La facilité de programmation de PHP permet donc à un développeur, de bonne foi mais sans culture de la sécurité des systèmes d'information, de développer des applications attrayantes mais mal sécurisées ou difficiles à sécuriser. Ce qui fait que les premières versions de nombreux projets développés dans ce langage sont souvent vulnérables. Parfois le développement ne va pas au dela de ces premières versions.

L'exemple typique de ce type d'application non maintenue et largement exploitée dans des attaques est l'outil `ExtCalendar`. Le CERTA a émis une alerte concernant cette application et, à la date de rédaction de cette note, il n'y a toujours pas de correctif de sécurité officiel.

Le code de ces applications plus ou moins sûres est ensuite fréquemment repris dans d'autres logiciels. Si bien que lorsqu'une vulnérabilité est corrigée dans le code d'origine, elle reste encore dans ses copies.

– PHP est un langage populaire. De nombreux développeurs ou contributeurs plus ou moins proffessionels offrent une gamme très étendue d'applications clés en main. De fait les applications écrites en PHP sont largement déployées. Une vulnérabilité risque donc d'affecter de nombreuses victimes.

  Un intrus qui aurait pour ambition de « faire du chiffre » en se créant un tableau de chasse de sites qu'il a compromis pourrait être interressé par une technologie largement répandue. Des sites consacrés à la revendication des défigurations (une forme d'intrusion destinée à changer, ajouter ou supprimer des pages) de sites WEB laissent penser qu'une telle motivation existent chez certains intrus.

  Quand une menace existe, il est primordial de réduire la vulnérabilité et les chemins d'attaques pour réduire le risque.

– PHP est un langage interprété. Le logiciel qui interprète les programmes peut être configuré pour offrir une très grande souplesse de programmation : il n'est pas nécessaire de déclarer des variables et de leur attribuer une valeur par défaut, les variables sont créées dès qu'elles sont invoquées la première fois. Cette facilité est couplée à une deuxième propriété des sites WEBs développés avec PHP : le visiteur du site peut dans certains cas définir (avant que le programme ne le fasse explicitement) une variable. Cette configuration a été pendant longtemps la configuration par défaut.

– PHP permet une utilisation transparente du réseau. Ainsi les mêmes fonctions permettent d'ouvrir un fichier local ou distant accessible par un protocole tel que HTTP ou FTP. Cette puissance du langage en fait une de ses principales vulnérabilités lorsqu'elle n'est pas maîtrisée et ouvre la porte à des nombres attaques dites de « PHP include ».

– Indépendamment des fonctionnalités dangereuses par défaut, l'interpréteur PHP a un lourd passé ayant obligé à la corrections de nombreuses vulnérabilités. Cela pourrait être le signe que la sécurité n'a pas été prise en compte à l'origine du développement de ce logiciel.

– les pages d'un site WEB mis en œuvre à l'aide d'une application écrites en PHP sont très reconnaissables dans la mesure où elles conservent le même nom d'une application à l'autre. Souvent le nom de l'application est écrit dans les pages présentées à l'utilisateur. Si bien qu'avec un moteur de recherche il est très facile de trouver des sites vulnérables.

Ces facteurs de risques ne sont pas que théoriques. Le CERTA a pu constater que les mêmes causes étaient bien souvent à la source des mêmes compromissions, en particulier des conjonctions de mauvaises pratiques aboutissant aux attaques de type « PHP include ». Les machines attaquées sont bien souvent découvertes par l'intrus par une simple réponse de moteur de recherche.

Ce tableau rapidement brossé du langage PHP peut paraître sombre. Il est cependant possible de développer et de déployer des applications raisonnablement sûres écrites en PHP. Pour atteindre cet objectif, il est *impératif* de prendre un certain nombre de précautions.

Ce document précise les conditions nécessaires pour qu'une application PHP puisse prétendre s'exécuter avec un minimum de sécurité.

3 Politique du CERTA vis-à-vis de PHP

Le CERTA assure une veille technique sur les failles de sécurité des applications. Les failles de sécurités découvertes puis corrigées ou non dans les applications écrites dans le langage PHP sont quotidiennes. Beaucoup d'applications sont vulnérables ; il n'est pas possible de publier tous les avis de sécurité pour l'ensemble des applications.

En particulier, de nombreuses applications à l'avenir incertain n'offrent pas le niveau de qualité minimum pour prétendre à un déploiement sur un site institutionnel. Ce sont en général des *gadgets* qui n'offrent que des fonctionnalités accessoires (calendrier, ...).

Le CERTA ne fera pas nécessairement d'avis de sécurité sur les vulnérabilités de ces logiciels à moins d'avoir la conviction qu'ils soient utilisées dans les administrations.

4 Conseils pour choisir une application

La plupart des critères de choix d'une application ne sont pas spécifiques au langage PHP. Mais le contexte dans lequel sont développées certaines applications avec ce langage doivent conduire à un surcroît de prudence.

- Il est primordial de choisir une application qui a un processus de maintenance. Celui-ci peut être interne, externalisé ou réalisé par la communauté des développeurs.
  Toutes les applications ont des failles. L'important n'est pas de chercher une application exempte de bogue, ce qui constitue un but illusoire, mais plutôt de s'assurer qu'en cas de problème, il y aura une maintenance dans un délais raisonnable.
  À ce titre, il convient de remarquer que parmi les *gadgets* écrits en PHP, un certain nombre ont des problèmes de sécurité depuis des années et ne sont toujours pas corrigés.
- Acquérir le logiciel dans un « contexte officiel ».
  La notion de contexte officiel est assez ouverte. L'attention du lecteur est attirée sur les problèmes, dans le cas de logiciels aux sources ouvertes, de l'acquisition de logiciels sur des serveurs mirroirs. Dans le passé, des sites mirroirs compromis ont fourni des applications PHP attrayantes, modifiées à l'insu de leurs auteurs pour présenter une porte dérobée.
  C'est le cas du logiciel `wordpress`. Ce logiciel destiné à mettre en œuvre des journaux WEBs (*blogs*) a été victime d'une attaque sur un des sites de distributions (cf. [**?**]). Ce n'est pas la qualité du logiciel qui est en cause, mais plus probablement sa popularité qui a motivé cette attaque.
  Un autre exemple de contexte non officiel est la mise à disposition d'une correctif sous la forme d'une remarque dans un forum de développeurs. C'est peut être utile dans une phase de recherche de la portée exacte de la vulnérabilité de communiquer de cette façon auprès de béta testeurs. Cela ne constitue en aucune façon un correctif définitif.
- Mettre en place un service de maintenance pour l'application. Pour un logiciel libre, il faut au minimum s'abonner à la liste de diffusion des correctifs de sécurité pour cette application (ou pour la distribution qui fournit cette application) et appliquer ces correctifs.
- Informer le CERTA. Il se peut que la politique de veille du CERTA vis-à-vis des applications développées en PHP ne couvre pas l'application choisie. Le CERTA dans le cadre du traitement d'incidents peut avoir connaissances d'attaques, de vulnérabilités, de risques qui affectent cette application.
  Ne pas informer le CERTA, c'est se priver de cette information.
- ne pas hésiter à faire procéder à un audit de code avant de déployer.

5 Conseils pour déployer et exploiter une application

Lors du déploiement d'une application PHP sur un serveur WEB, il y a quelques paramètres de configuration qui permettent d'éviter de nombreux problèmes.

5.1 La variable `register_globals`

La première chose à faire pour se prémunir des conséquences de la souplesse offerte par PHP pour que des variables soient créées automatiquement, éventuellement à l'insu du développeur, est de modifier la valeur de la variable `register_globals`. Cette variable peut prendre les valeurs `on` ou `off`. Pour garantir un minimum de séurité il faut *impérativement* que cette valeur prenne la valeur `off`.

Ce réglage se fait dans les fichiers `php.ini`. Il peut y avoir plusieurs fichiers de ce type (par exemple pour prendre en compte plusieurs versions de php). Ces fichiers doivent contenir la ligne suivante :

```
register_globals = Off
```

5.2 La variable `safe_mode`

Le langage PHP dispose d'un mode de fonctionnement dans lequel il est possible de configurer finement la sécurité en limitant les fonctions dangereuses. Il est conseillé de lire la documentation de ce mode (cf. [**?**]) et de configurer le système afin d'être le moins permissif possible.

En particulier, il est sain de veiller à :

- restreindre les possibilités de lecture strictement aux répertoires nécessaires au bon fonctionnement de l'application afin d'éviter qu'une faille ne permette de lire des fichiers sensibles comme des fichiers de configurations du système, des données personnelles ou des journaux de connexions ;

– à restreindre les fonctions utilisables. En particulier les fonctions de PHP qui permettent de lancer des commandes devraient être désactivées si l'application ne s'en sert pas.

5.3 Mettre en œuvre un pare-feu

Les attaques basées sur l'inclusion de fichiers PHP distants (dites « PHP include ») sont très fréquentes. Dans ce genre d'attaque l'intrus force le serveur WEB vulnérable à télécharger un fichier sur l'Internet. Ce fichier contient du code PHP, qui est exécuté automatiquement par l'interpréteur PHP de la machine vulnérable, conduisant ainsi à l'exécution de code arbitraire.

Le pare-feu devrait être configuré pour interdire les connexions sortantes depuis le serveur WEB qui accueille l'application PHP. Idéalement ce pare-feu ne doit pas être un pare-feu installé sur le serveur WEB mais un pare-feu en coupure. Plus d'information dans la note [**?**].

5.4 Filtrer les requêtes

Certains outils (*reverse proxy*) permettent d'écarter certaines requêtes faites sur le site (par exemple parce qu'elles demandes des chemins qui n'existent pas). De tels outils, bien qu'imparfaits, réduisent l'exposition d'un site fragile aux agressions de l'Internet.

5.5 Journaliser

PHP peut être configuré pour journaliser toutes les erreurs d'exécution. S'il n'est pas possible d'empêcher les attaques, il peut être intéressant d'augmenter le niveau de bruits qu'elles produisent pour les découvrir au plus tôt. Pour cela tous les programmes devraient commencer par la ligne :

```
<?php error_reporting(E_ALL);>
```

Bien entendu, la journalisation en elle-même ne résoud aucun problème. Elle permet si les journaux sont dépouillés de découvrir les problèmes ce qui est déjà beaucoup. Il est donc important d'affecter des ressources au dépouillement de ces journaux.

5.6 Attention à l'hébergement mutualisé

Si vous optez pour un hébergement mutualisé ou tout autre forme de sous-traitance, il faut préciser contractuellement les exigences de ce chapitre, avec au besoin des pénalités.

Certains hébergeurs offrent une grande souplesse au détriment de la sécurité (en particulier sur les points ci-dessus). Il peut s'agir d'un critère de choix des hébergeurs : être à même d'offrir ces paramètres de sécurité.

6 Conseil au développeurs

6.1 Faire de la programmation défensive

La programmation défensive n'est pas une notion propre à PHP. C'est un état d'esprit que doit acquérir le développeur qui souhaite avoir une application plus résitante aux attaques.

Le typage fort n'existe pas dans ce langage de programmation. Il est donc nécessaire de faire *à la main* certains contrôles de cohérence qui sont fait automatiquement dans d'autre langages.

6.1.1 Filtrer les entrées

De nombreuses attaques sur des sites qui utilisent PHP mettent en œuvre de l'injection de données (injection de sql, injection de commandes shell, *cross site scripting*, ...). Ces attaques sont possibles parce que les entrées de données sur lesquelles le visiteur peut avoir une influence ne sont pas « nettoyées » avant d'être passées en paramètres à des fonctions du système.

Le développeur est invité à nettoyer ses variables afin de vérifier qu'elles ne contiennent que ce qu'elles sont sensées contenir. Par exemple un nombre ne doit contenir que des chiffres (et pas des morceaux d'URL ou de requêtes SQL). Un identifiant ne devrait pas contenir des caractères délimitant les chaînes de caractères comme « `"` » ou « `'` ».

6.1.2 Vérifier la cohérence des données

Le programmeur d'une application qui doit tourner dans un milieu hostile ne doit pas ternir pour acquis que la cohérence des données va se maintenir toute seule.

Le programmeur est invité à vérifier régulièrement la cohérence :

- des types de données ;
- des tailles de tableau ;
- des conditions d'entrées ou de sorties d'une fonction ou d'un module.

6.1.3 N'utiliser que des flots de données explicites

Les variables globales sont nuisibles dans la mesure où elles introduisent des effets de bord. En particulier elles permettent souvent une communication entre différentes partie du code. Une communication par ce biais est difficile à maîtriser, il est recommandé d'éviter d'avoir recours à ce genre moyen.

Il existe une version modifiée de PHP (`hardened php`) qui supprime certaines variables globales. Il est prudent de développer comme si cette extension n'était pas installée et de recommander de l'installer.

6.2 Procédure d'installation

De nombreux utilisateurs d'applications PHP les installent dans un environement peu sûr. Alors que l'application est développée avec grand soin, l'environnement d'exécution peut être faible. Cela peut entâcher à tort la réputation de l'application.

La procédure d'installation d'une application PHP peut utilement vérifier la configuration (`register_globals`, ...) et proposer des modifications de la configuration pour la rendre plus sûre.

Gestion détaillée du document

20 mars 2007 version initiale.