Sécurité et PHP : Mission impossible ?

Passons un peu la tête au dessus du guidon…

Les discussions autour du manque de sécurité dans PHP ne manquent pas sur la toile. Pourtant, à y regarder de plus près, ces prétendues "failles" sont souvent issues de mauvaises pratiques de développement ou du piètre niveau de sécurité de certains composants extérieurs. Alors, impossible de coder une application sécurisée en PHP ? Pas si sûr...

On n'en finit plus de décrier le manque de sécurité dans PHP. Les démonstrations techniques [1] prouvant par A+B qu'il est possible de provoquer des buffers overflow et autres injections en tout genre pullulent sur Internet, à grand coup d'extraits de (l'infâme) code source PHP et de hexdump de pages mémoire. Il suffit de consulter le site de l'équipe du « Hardened-PHP Project » [2] pour se rendre compte de l'impressionnante liste des vulnérabilités affectant ce langage de débutant (PHP a initialement été conçu pour être facilement maîtrisé et pour coller aux besoins, sans se soucier de la sécurité).

Il ne s'agit pas de remettre en cause ces présentations réellement passionnantes, pour certaines. Mais démontrer la présence de vulnérabilités avec « *Register_globals=On* » ou à l'aide de scripts avec des erreurs de développement manifestes, c'est un peu facile. Dans ce cas, en effet, il est trivial de démontrer que le suivi de session sous PHP ne vaut rien ou que les sites développés sous ce langage sont bizarrement victimes d'une pandémie de Remote File Inclusion. Même Wikipédia s'y met, en associant Remote File Inclusion [3] au langage PHP !

Oui, PHP souffre intrinsèquement de failles de sécurité pouvant parfois conduire à de l'exécution de code. C'est la conséquence d'une disparité dans la qualité du code source des différents modules. Mais au-delà des failles intrinsèques au langage, il y a surtout les failles liées au développement lui-même. PHP n'est certainement pas, à lui seul, un échec.

[La sécurité par l'obscurité est un échec]

Et .NET alors ? On ne sait pas, on n'a pas les sources. Mais en tout cas c'est beaucoup mieux que PHP. Ok, donc jusqu'ici, tout va bien, c'est rassurant !

Pas vraiment, parce que tout le monde sait que le concept de la sécurité par l'obscurité est un échec total ; n'hésitons pas à le rappeler !

[Mauvais langage ou mauvaise pratique ?]

Mais on s'égare, revenons à nos moutons.

En local, il y a des failles exploitables dans PHP, c'est indéniable et d'une facilité déconcertante. En remote, c'est déjà bien plus compliqué, si l'on s'entend bien sur le fait que l'application doit être exempte de failles de sécurité liées à un mauvais développement.

Oui, *include ($_GET['file']) ;* c'est un exemple de très mauvais développement.

Ça tombe plutôt bien car il faut être motivé pour décider de coder une application native pour Linux ou Windows à base de PHP. Ne souriez pas, il semblerait qu'il existe des personnes ultra-motivées qui proposent, depuis PHP, un accès direct aux interfaces de l'API Windows [4]. Ca fait froid dans le dos…

PHP est victime de son succès et du manque de maîtrise générale autour de l'utilisation de ce langage. Mais cette remarque est probablement vraie pour la quasi-totalité des langages de développement, peu de personnes codent vraiment à l'état de l'art (d'ailleurs, c'est quoi l'état de l'art du développement ?).

[Kevin Mitnick hacké, où va t'on…]

PHP est également, pour ces raisons de simplicité, l'un des langages de développement les plus utilisés sur les plateformes d'hébergement Web. Plus un langage est populaire, plus il est facile de trouver des failles de sécurité dans les applications développées avec. Alors, y aurait-il un rapport entre plateforme d'hébergement mutualisé et faille de sécurité ? Absolument, si on regarde les « célébrités du milieu » (Kevin Mitnick, Dan Kaminsky et Julien Tinnes entre autres) qui se sont faits « hacker », soi-disant victimes du piètre niveau de sécurité de leurs hébergeurs [5].

Le site de Kevin Mitnick a beau être développé en PHP, c'est par la grande porte laissée ouverte, avec les clés sous le paillasson, que les pirates sont entrés. PHP n'y est pour rien, par contre côté bonne pratique de sécurité on est loin du compte (hint : *PermitRootLogin*, mots de passe complexes, politique de filtrage…).

[1] http://www.blackhat.com/presentations/bh-usa-09/ESSER/BHUSA09-Esser-PostExploitationPHP-SLIDES.pdf

[2] http://www.php-security.org/

[3] http://en.wikipedia.org/wiki/Remote_File_Inclusion

[4] http://winbinder.org/

[5] http://www.wired.com/threatlevel/2009/07/kaminsky-hacked/

[Toujours pas de traitement contre le CNI]

Mais alors, les problématiques de sécurité sur Internet en général, dans PHP en particulier, ne seraient pas exclusivement liées au langage de développement ? Cela voudrait-il dire que la sécurité du système d'exploitation et du serveur Web sous-jacent sont également à prendre en compte ?

Mais alors, en compilant correctement PHP et (au hasard) Apache il serait possible d'améliorer le niveau de sécurité des applications Web ?

C'est vrai qu'en désactivant toutes les fonctionnalités et les modules inutiles, il serait possible de réduire la surface d'attaque et une bonne partie des failles de sécurité. Mais pour ça il faut auparavant prendre un traitement efficace contre le *syndrome CNI* (« Click », « Next », « Install ») qui semble encore avoir de beaux jours devant lui (installation par défaut = danger).

Attention également à ne pas tomber dans l'excès inverse. C'est généralement une bonne pratique que de ne pas chercher à réinventer la roue à chaque fois. Un Framework de développement, sous réserve de l'utiliser correctement, peut ainsi contribuer à améliorer le niveau de sécurité des applications.

A me lire, on va même finir par penser qu'il serait possible de sécuriser les serveurs mutualisés… Bon ok, ce n'est pas gagné ! Même si des solutions existent et gagnent à être connues…

Mais en parlant de vraie solution, est-il pertinent d'utiliser un serveur mutualisé pour héberger des données ultra-critiques ? Bien sûr que non, alors pourquoi autant de sociétés font ce choix ?

Ca doit être frustrant à force, de rechercher des solutions techniques pour des problèmes qui n'existeraient même pas si on ajoutait quelques grammes de bon sens dans les choix du quotidien…

C'est mission impossible que de faire le tour de la question en quelques lignes, il faudrait un livre.

Mais par pitié, sortons un peu la tête du guidon et n'oublions pas que **la sécurité n'est qu'un mythe.** Ce n'est pas un but à atteindre mais un chemin qu'il faut impérativement suivre.

PHP n'est que l'un des tous derniers maillons de la chaîne de l'insécurité. S'il fallait encore vous convaincre, jetez un œil à la page 96 du magazine Capital de juillet 2009 [6]. C'est consternant !

[Conclusion]

Si PHP était ultra-sécurisé, il serait beaucoup moins accessible et les développeurs se tourneraient vers d'autres langages.

C'est le manque de maîtrise et de sensibilisation aux bonnes pratiques qui nuisent fortement au niveau de sécurité. PHP n'est pas pire que ses homologues interprétés, compilés ou pseudo compilés.

C'est du moins ce qu'on constate au hasard d'un test d'intrusion ou d'un audit de code source…

Jérémie Jourdin – jeremie.jourdin@advens.fr

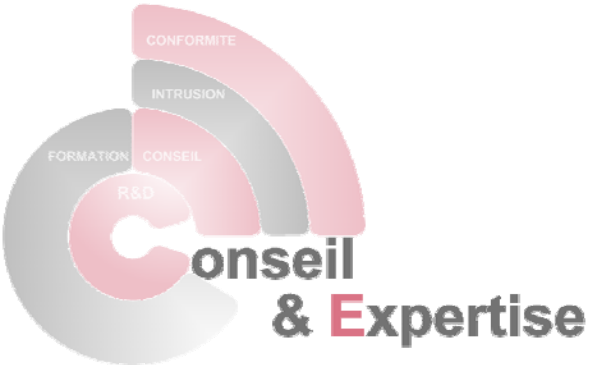
Responsable pôle « Conseil & Expertise »

Le Pôle « Conseil & Expertise » concentre le savoir-faire technique d'Advens.

Nous accompagnons nos clients dans leurs problématiques de sécurité opérationnelles et les aidons à qualifier leur niveau de sécurité.

Nous intervenons dans le choix des technologies de protection adaptées aux enjeux de sécurité propres à chacun.

Forts de notre veille technologique et de notre centre R&D, nous concevons également des solutions de sécurité innovantes.



[6] http://www.capital.fr/le-magazine/magazine-n-215