Cours de sécurité accéléré no. 1 - Never trust foreign data

Cours de sécurité accéléré no. 1 - Never trust foreign data

Cet article a été écrit par Damien Metzger, et publié sur le blog de PrestaShop le 21 mai 2011.

Si une seule règle doit être retenue par un développeur en matière de sécurité, c'est bien celle-ci : « Never trust foreign data », ou encore « Ne jamais faire confiance aux données venant de l'extérieur ». Voyons ensemble dans le détail ce que cet adage des temps moderne signifie :

- **Never** : dans le domaine de la sécurité, il ne faut pas viser plus bas que la perfection. Un code sécurisé à 99% est un code qui comporte des failles. Un pirate n'a en effet pas besoin de grand-chose pour agir. Une fois la faille ou le point d'entrée trouvé, il pourra bien souvent étendre son pouvoir et poser des backdoors.
- **Trust** : quand on dit qu'il ne faut pas faire confiance, je dirais plus exactement qu'il faut considérer toute donnée comme malicieuse. Partez toujours du principe que la variable que vous utilisez a été renseignée par un pirate, et vous ne serez jamais pris au dépourvu. Inutile de réfléchir à comment il peut faire ça et toujours partir du principe qu'il peut le faire.
- **Foreign** : la définition de donnée « extérieure » doit être très stricte. En règle générale, toute variable qui n'a pas été assignée au sein de la fonction dans laquelle elle est utilisée doit être considérée comme étrangère. Même si c'est vous-même qui l'avez passée en paramètre quelques minutes auparavant, vous ne savez pas quel autre développeur pourra modifier votre code ultérieurement, ni même si vous vous souviendrez parfaitement de ce que vous aviez fait dans quelques mois.
- **Data** : une donnée est généralement une variable. Passée en paramètre, en POST ou GET, provenant d'un cookie, ou même provenant de $_SERVER (qui n'est pas moins étranger que le reste !). Mais une donnée peut également être le contenu d'un fichier –ou même le chemin vers ce fichier-, la réponse d'un web service, les en-têtes des requêtes HTTP et bien d'autres encore. Tout ce qui n'est pas directement affiché dans votre éditeur de code préféré est une donnée.

Il faut donc retenir que rien de ce que vous utilisez n'est fiable. L'avantage d'avoir tout le temps cette idée à l'esprit, c'est qu'elle devient très rapidement naturelle. Il devient alors inimaginable de ne pas fonctionner de cette manière.

Quelle est la solution alors ? Que doit-on faire de nos données, si on ne peut plus les utiliser comme on le souhaite ? Réponse en 2 parties :

- Faites des contrôles. Si vous attendez un entier, alors vérifiez que c'est bien un entier que vous utilisez. Si c'est un hash MD5, alors il ne doit rien y avoir d'autre que des caractères alphanumériques. Faites des expressions régulières, liste blanche quand c'est possible, liste noire quand ça ne l'est pas. Attention tout de même, les regexps sont gourmandes en ressources CPU, il donc privilégiez dans la mesure du possible des fonctions plus simple comme is_numeric() ou strstr(), ou encore is_array() ou is_object().
- Faites des casts. Avec un cast, vous êtes sûrs à 100% du type de donnée que vous utilisez. C'est déjà un pas en avant. Bien évidemment, un cast en string ne résoudra pas vos problèmes comme par magie, le contrôle préalable est donc toujours primordial.