Communiqué Sécurité et fonction de hash des mots de passe

Lire ceci :

- https://stackoverflow.com/a/29148605
- https://www.php.net/manual/en/faq.passwords.php
- https://www.securityfocus.com/blogs/262

Donc définitivement il faut prendre password_hash( ) et password_verify( ) !

Autres lecture intéressantes:

- https://stackoverflow.com/questions/2235158/sha1-vs-md5-vs-sha256-which-to-use-for-a-php-login

**** N'oubliez pas que même si vos données sont d'une importance non capitale, la compromission des mots de passe utilisateurs peut menacer leurs autres comptes puisqu'un même utilisateur peut parfois réutiliser des login ou des variantes sur plusieurs sites. C'est donc une question d'éthique et de savoir-être informatique de bien crypter les mots de passes avec la meilleure technologie disponible et permise.

==========

Extrait de la doc officielle de php.net

How should I hash my passwords, if the common hash functions are not suitable?

When hashing passwords, the two most important considerations are the computational expense, and the salt. The more computationally expensive the hashing algorithm, the longer it will take to brute force its output.

PHP 5.5 provides a native password hashing API that safely handles both hashing and verifying passwords in a secure manner. There is also » a pure PHP compatibility library available for PHP 5.3.7 and later.

Another option is the crypt() function, which supports several hashing algorithms in PHP 5.3 and later. When using this function, you are guaranteed that the algorithm you select is available, as PHP contains native implementations of each supported algorithm, in case one or more are not supported by your system.

The suggested algorithm to use when hashing passwords is Blowfish, which is also the default used by the password hashing API, as it is significantly more computationally expensive than MD5 or SHA1, while still being scalable.

Note that if you are using crypt() to verify a password, you will need to take care to prevent timing attacks by using a constant time string comparison. Neither PHP's == and === operators nor strcmp() perform constant time string comparisons. As password_verify() will do this for you, you are strongly encouraged to use the native password hashing API whenever possible.