password_hash

(PHP 5 >= 5.5.0, PHP 7)

password_hash — Crea un hash de contraseña

Descripción ¶ ¶

string **password_hash** (string `$password` , integer `$algo` [, array `$options`])

password_hash() crea un nuevo hash de contraseña usando un algoritmo de hash fuerte de único sentido. **password_hash()** es compatible con crypt(). Por lo tanto, los hash de contraseñas creados con crypt() se pueden usar con **password_hash()**.

Actualmente se admiten los siguientes algoritmos:

- **`PASSWORD_DEFAULT`** - Usar el algoritmo bcrypt (predeterminado a partir de PHP 5.5.0). Observe que esta constante está diseñada para cambiar siempre que se añada un algoritmo nuevo y más fuerte a PHP. Por esta razón, la longitud del resultado de usar este identificador puede cambiar con el tiempo. Por lo tanto, se recomienda almacenar el resultado en una columna de una base de datos que pueda apliarse a más de 60 caracteres (255 caracteres sería una buena elección).
- **`PASSWORD_BCRYPT`** - Usar el algoritmo **`CRYPT_BLOWFISH`** para crear el hash. Producirá un hash estándar compatible con crypt() utilizando el identificador "$2y$". El resultado siempre será un string de 60 caracteres, o **`FALSE`** en caso de error.

  Opciones admitias:

  - *salt* - para proporcionar manualmente una sal a usar cuando se realiza el hash de la contraseña. Observe que esto sobrescribirá y prevendrá que una sal sea generada automáticamente.

    Si se omite, se generará una sal aleatoria mediante **password_hash()** para cada hash. Este es el modo de operación intencionado.

    Advertencia

    La opción salt está obsoleta a partir de PHP 7.0.0. Ahora se prefiere simplemente utilizar generada de manera predeterminada.

  - *cost* - denota el coste del algoritmo que debería usarse. Se pueden encontrar ejemplo de estos valores en la página de crypt().

    Si se omite, se usará el valor predeterminado *10*. Este es un buen coste de referencia, pero se podría considerar aumentarlo dependiendo del hardware.

Parámetros ¶ ¶

`password`

La contraseña del usuario.

Precaución

El uso de **`PASSWORD_BCRYPT`** como el algoritmo resultará en el truncamiento del parámetro `password` a un máximo de 72 caracteres de longitud.

`algo`

A constante del algoritmo de contraseñas indicando qué algoritmo utilizar para crear el hash de la contraseña.

`options`

Un array asociativo de opciones. Véanse las constantes de algoritmos de contraseñas para la documentación sobre las opociones admitidas de cada algoritmo.

Si no se indica, se creará una sal aleatoria y el coste algorítmico por defecto será utilizado.

Valores devueltos ¶ ¶

Devuelve el hash de la contraseña, o **`FALSE`** en caso de error.

El algoritmo, coste y sal usados son devueltos como parte del hash. Por lo tanto, toda la información que es necesaria para verificar el hash, está incluida en él. Esto permite que la función password_verify() verifique el hash sin tener que almacenar por separado la información de la sal o del algoritmo.

Ejemplos ¶ ¶

Ejemplo #1 Ejemplo de password_hash()

```php
<?php
/**
 * Queremos crear un hash de nuestra contraseña uando el algoritmo DEFAULT actual.
 * Actualmente es BCRYPT, y producirá un resultado de 60 caracteres.
 *
 * Hay que tener en cuenta que DEFAULT puede cambiar con el tiempo, por lo que debería prepararse
 * para permitir que el almacenamento se amplíe a más de 60 caracteres (255 estaría bien)
 */
echo password_hash("rasmuslerdorf", PASSWORD_DEFAULT)."\n";
?>
```

El resultado del ejemplo sería algo similar a:

```
$2y$10$.vGA1O9wmRjrwAVXD98HNOgsNpDczlqm3Jq7KnEd1rVAGv3Fykk1a
```

Ejemplo #2 Ejemplo de password_hash() estableciendo el coste manualmente

```php
<?php
/**
 * En este caso, queremos aumentar el coste predeterminado de BCRYPT a 12.
 * Observe que también cambiamos a BCRYPT, que tendrá siempre 60 caracteres.
 */
$opciones = [
    'cost' => 12,
];
echo password_hash("rasmuslerdorf", PASSWORD_BCRYPT, $opciones)."\n";
?>
```

El resultado del ejemplo sería algo similar a:

```
$2y$12$QjSH496pcT5CEbzjD/vtVeH03tfHKFy36d4J0Ltp3lRtee9HDxY3K
```

password_verify

(PHP 5 >= 5.5.0, PHP 7)

password_verify — Comprueba que la contraseña coincida con un hash

Descripción ¶

boolean **password_verify** (string `$password` , string `$hash`)

Comprueba que el hash proporcionado coincida con la contraseña facilitada.

Observe que password_hash() devuelve el algoritmo, el coste y la sal como parte del hash devuelto. Por lo tanto, toda la información que es necesaria para verificar el hash está incluida. Esto permite a la función de verificación comprobar el hash sin la necesidad de almacenar por separado la información de la sal o del algoritmo.

Parámetros ¶

`password`

La contraseña del usuario.

`hash`

Un hash creado por password_hash().

Valores devueltos ¶

Devuelve **`TRUE`** si la contraseña y el hash coinciden, o **`FALSE`** de lo contrario.

Ejemplos ¶

Ejemplo #1 Ejemplo de password_verify()

```
<?php
// Ver el ejemplo de password_hash() para ver de dónde viene este hash.
$hash = '$2y$07$BCryptRequires22Chrcte/VlQH0piJtjXl.0t1XkA8pw9dMXTpOq';

if (password_verify('rasmuslerdorf', $hash)) {
    echo '¡La contraseña es válida!';
} else {
    echo 'La contraseña no es válida.';
```

```
}
?>
```

El resultado del ejemplo sería:

```
¡La contraseña es válida!
```

Ver también ¶

- password_hash() - Crea un hash de contraseña
- » implementación en el espacio de usuario

add a note

User Contributed Notes 4 notes

up
down
129
Anonymous ¶
2 years ago

If you get incorrect false responses from password_verify when manually including the hash variable (eg. for testing) and you know it should be correct, make sure you are enclosing the hash variable in single quotes (') and not double quotes (").

PHP parses anything that starts with a $ inside double quotes as a variable:

```
<?php
// this will result in 'Invalid Password' as the hash is parsed into 3 variables of
// $2y, $07 and $BCryptRequires22Chrcte/VlQH0piJtjXl.0t1XkA8pw9dMXTpOq
// due to it being enclosed inside double quotes
$hash = "$2y$07$BCryptRequires22Chrcte/VlQH0piJtjXl.0t1XkA8pw9dMXTpOq";

// this will result in 'Password is valid' as variables are not parsed inside single quotes
$hash = '$2y$07$BCryptRequires22Chrcte/VlQH0piJtjXl.0t1XkA8pw9dMXTpOq';

if (password_verify('rasmuslerdorf', $hash)) {
    echo 'Password is valid!';
} else {
    echo 'Invalid password.';
}
?>
```

up
down
31
Vasil Toshkov ¶
3 years ago

This function can be used to verify hashes created with other functions like crypt(). For example:

```
<?php

$hash = '$1$toHVx1uW$KIvW9yGZZSU/1YOidHeqJ/';

if (password_verify('rasmuslerdorf', $hash)) {
    echo 'Password is valid!';
} else {
    echo 'Invalid password.';
}
```

```
// Output: Password is valid!

?>
```

up
down
6
chris at weeone dot de ¶
10 months ago

The function password_verify() uses constant time. This makes it safe against timing attacks. Don't use crypt($password_database) === crypt($password_given_by_login), since there is no protection against timing attacks!

If you don't want to use password_verify(), then have a look at hash_equals(), which also runs a timing attack safe string comparison.

up
down
4
suit at rebell dot at ¶
3 years ago

As Vasil Toshkov stated, password_verify() can be used to verify a password created by crypt() or password_hash()

That is because passwords created by password_hash() also use the C crypt sheme

If you want to verify older plain MD5-Hashes you just need to prefix them with 1

See https://en.wikipedia.org/wiki/Crypt_(C) for more information.

add a note

- Funciones de hashing de contraseñas
  - password_get_info
  - password_hash
  - password_needs_rehash
  - password_verify

- My PHP.net
- Contact
- Other PHP.net sites
- Mirror sites
- Privacy policy