Poprawne przetrzymanie hasła w PHP

v 1.1

Coders Lab
SZKOŁA PROGRAMOWANIA

Hasła nie trzymamy w bazie danych!

Bazy danych nie są w 100% bezpieczne. Dlatego nigdy nie powinniśmy trzymać w bazie hasła, które wpisał użytkownik.

W bazie danych powinniśmy trzymać tylko i wyłącznie zasolone hasło (salted password).

Co to jest sól kryptograficzna?

- Sól kryptograficzna jest dodatkową informacją przyjmowaną przez algorytm haszujący.
- Dzięki takiemu rozwiązaniu nasze hasło staje się trudniejsze do złamania przy użyciu standardowych technik (np. tęczowych tablic).

- Niewskazane jest używanie takiej samej soli do haszowania wszystkich haseł w naszej aplikacji, co daje taki sam wynik jak nieużywanie żadnej soli.

Najlepiej dla każdego hasła wygenerować i przetrzymywać unikatową wartość soli.

Funkcje kryptograficzne i algorytmy haszujące

Funkcje

PHP (od wersji 5.5.0) implementuje podstawowe funkcje kryptograficzne do zabezpieczania haseł:

- **password_get_info**
- **password_hash**
- **password_needs_rehash**
- **password_verify**

Algorytmy

Spośród wielu algorytmów haszujących warto wskazać najpopularniejsze:

- **MD5**
- **SHA1**
- **SHA256**
- **Blowfish**

password_hash

password_hash jest podstawową funkcją służącą do haszowania naszych haseł.

Przyjmuje ona jako dane wejściowe:

- nasze hasło,
- typ algorytmu, jaki ma być użyty,
- dodatkowe opcje.

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

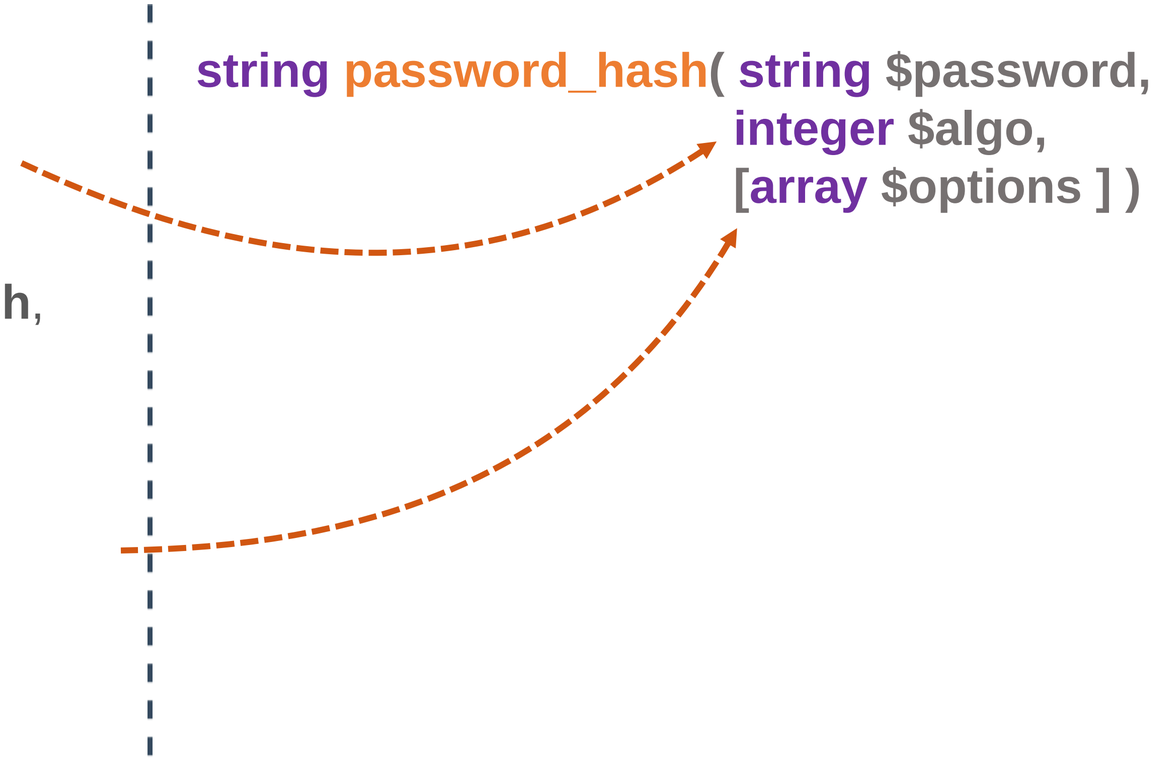
password_hash

Parametr $algo

Może przyjąć – w uproszczeniu – dwie opcje:

- **PASSWORD_DEFAULT**,
- **PASSWORD_BCRYPT**.

W obu opcjach jest stosowany algorytm **blowfish**, który zawsze zwróci nam hasło o długości 60 znaków.

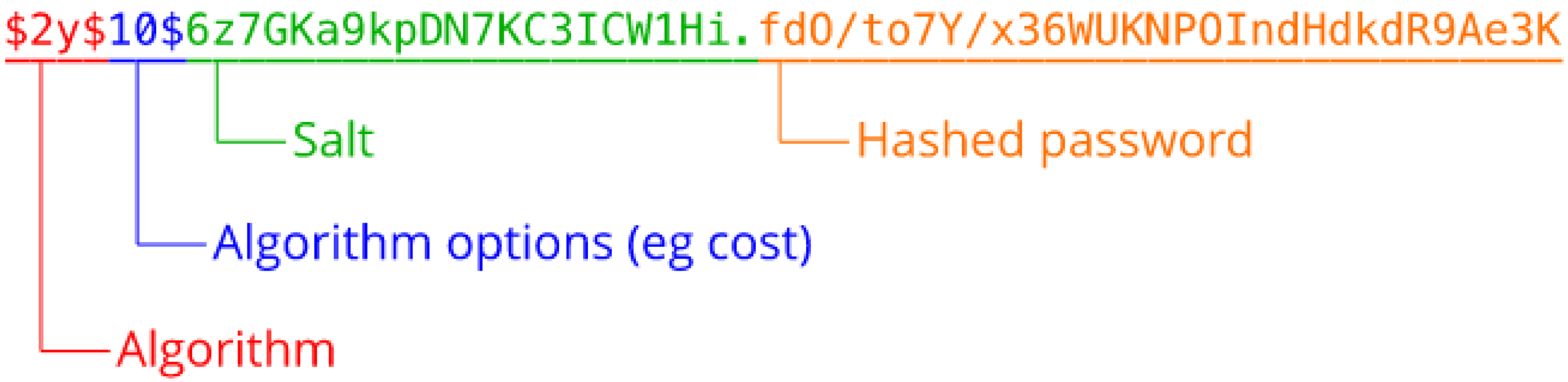


Parametr $options

Jest to tablica, która może mieć dwie komórki:

- **salt** – wartość soli, którą chcemy użyć,
- **cost** – koszt naszego haszowania (im większa długość trwania tym hasło jest bezpieczniejsze).

Jak trzymana jest nasza sól?



Stworzenie bezpiecznego hasła

```
$options = [
    'cost' => 11
];

$hashedPas = password_hash($pass, PASSWORD_BCRYPT, $options);
```

password_verify

Mamy zapisane w bazie danych zasolone hasło (razem z solą).

Jak możemy sprawdzić, czy hasło, które wpisał użytkownik, jest tym samym co to zapisane w naszej bazie?

password_verify jest funkcją sprawdzającą nasze hasło. Jeżeli hasło się zgadza zwraca TRUE, jeśli nie – FALSE.

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

Podstawowe błędy tworzenia haseł

- Nieużywanie losowej soli.
- Przesyłanie hasła w plain text.
- Używanie przestarzałych funkcji haszujących.
- Używanie dziwnych kombinacji funkcji haszujących (np. MD5(SH1(MD5(SH1(haslo))))).