Poprawne przetrzymanie hasła w Javie

v3.1

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

Spośród wielu algorytmów haszujących warto wskazać najpopularniejsze:

MD5

SHA1

SHA256

Blowfish

JBCrypt

`JBCrypt` jest implementacją algorytmu Blowfish, którą można ściągnąć z tej strony:

http://www.mindrot.org/projects/jBCrypt

Podstawowe metody tej implementacji to:

- **`checkpw(String plaintext, String hashed)`**. Sprawdza, czy hasło pasuje do hasha.
- **`gensalt()`**. Generuje sól.
- **`hashpw(String password, String salt)`**.

BCrypt.hashpw

`BCrypt.hashpw` jest metodą służącą do haszowania naszych haseł.

Przyjmuje ona jako dane wejściowe:

- nasze hasło,
- sól.

```
String password = "password";
String hashed = BCrypt.hashpw(password, BCrypt.gensalt());
```

BCrypt.checkpw

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?

`BCrypt.checkpw` jest metodą sprawdzającą nasze hasło. Jeżeli hasło się zgadza zwraca **`true`**, jeśli nie – **`false`**.

```
if (BCrypt.checkpw(candidate, hashed)) {
    System.out.println("Ok");
} else {
    System.out.println("It does not match");
}
```

BCrypt.checkpw

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?

`BCrypt.checkpw` jest metodą sprawdzającą nasze hasło. Jeżeli hasło się zgadza zwraca **`true`**, jeśli nie – **`false`**.

```
if (BCrypt.checkpw(candidate, hashed)) {
    System.out.println("Ok");
} else {
    System.out.println("It does not match");
}
```

`candidate` – hasło do sprawdzenia.

BCrypt.checkpw

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?

`BCrypt.checkpw` jest metodą sprawdzającą nasze hasło. Jeżeli hasło się zgadza zwraca **`true`**, jeśli nie – **`false`**.

```
if (BCrypt.checkpw(candidate, hashed)) {
    System.out.println("Ok");
} else {
    System.out.println("It does not match");
}
```

`candidate` – hasło do sprawdzenia.

`hashed` – zapisany wcześniej **`hash`**, do porównania.

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))))).