GABRIELE ROMANATO

Utilizzo di Bcrypt in PHP per la Sicurezza delle Password

Proteggere le password degli utenti è una delle responsabilità più importanti nello sviluppo di applicazioni web. PHP offre una soluzione sicura e moderna per la gestione degli hash delle password tramite l'algoritmo Bcrypt. In questo articolo vedremo come utilizzarlo correttamente.

Perché usare Bcrypt?

Bcrypt è progettato per essere computazionalmente costoso, rendendo difficile per un attaccante effettuare attacchi di forza bruta. È inoltre resistente ad attacchi con rainbow table, grazie all'utilizzo di un salt incorporato.

Hashing di una password

PHP semplifica l'hashing delle password tramite la funzione `password_hash()`, che utilizza Bcrypt per impostazione predefinita.

```
$password = 'miaPasswordSicura';
$hash = password_hash($password, PASSWORD_BCRYPT);
echo $hash;
```

Il valore restituito è una stringa che include anche il salt e il costo.

Verifica della password

Per verificare una password inserita con l'hash salvato, si utilizza `password_verify()`:

```
$input = 'miaPasswordSicura';
$savedHash = '$2y$10$abcdefghijk...';

if (password_verify($input, $savedHash)) {
    echo 'Password corretta';
} else {
    echo 'Password errata';
}
```

Aggiornamento dell'hash

Nel tempo, potrebbe essere utile aumentare il costo dell'hashing. La funzione `password_needs_rehash()` aiuta a determinare se è necessario rigenerare l'hash:

```
$options = ['cost' => 12];

if (password_needs_rehash($savedHash, PASSWORD_BCRYPT,
$options)) {
    $savedHash = password_hash($input, PASSWORD_BCRYPT,
$options);
}
```

Conclusione

Utilizzare Bcrypt in PHP è un modo efficace e sicuro per gestire le password degli utenti. Le funzioni native di PHP semplificano il processo, mantenendo un alto livello di sicurezza senza dover gestire manualmente salt o algoritmi complessi.