؟ سؤال ۱

تابع `password_hash()` چه کاری انجام می‌دهد و چرا هر بار خروجی متفاوت است؟

✅جواب:
این تابع رمز خام را به یک هش غیرقابل برگشت تبدیل می‌کند و برای امنیت بیشتر، خودش **salt**تصادفی می‌سازد. به همین دلیل حتی اگر یک رمز (مثلاً `123456`) را چند بار هش کنیم، خروجی هر بار فرق می‌کند.

؟ سؤال ۲

چرا نمی‌توانیم رمز خام کاربر را مستقیم با هش ذخیره‌شده مقایسه کنیم؟

✅جواب:
چون هش‌ها هر بار فرق می‌کنند و مساوی شدن مستقیم ممکن نیست.
مثال:

```
$hash = password_hash("123456", PASSWORD_DEFAULT);
echo "123456" === $hash ? "برابر" : "نابرابر"; // همیشه "نابرابر"
```

برای همین باید از `password_verify()` استفاده کنیم.

؟ سؤال ۳

کار تابع `password_verify()` چیست؟

✅جواب:
این تابع رمز خامی که کاربر وارد کرده را می‌گیرد، همان الگوریتم هش را روی آن اعمال می‌کند و با هش ذخیره‌شده مقایسه می‌کند.
اگر درست باشد `true` برمی‌گرداند، وگرنه `false`.

؟ سؤال ۴

کدام نوع ستون دیتابیس برای ذخیره هش مناسب است؟

✅**جواب:**
ستون از نوع `VARCHAR(255)` **یا** `TEXT` **کافی است. چون خروجی** `password_hash()` **معمولاً تا ۶۰ کاراکتر (bcrypt) یا کمی بیشتر (argon2) است.**

؟ سؤال ۵

چطور می‌توانیم بفهمیم هش ذخیره‌شده قدیمی است و باید دوباره ساخته شود؟

✅**جواب:**
با استفاده از تابع `password_needs_rehash()`.
مثال:

```
if (password_needs_rehash($hash, PASSWORD_DEFAULT,
['cost' => 12])) {
    $newHash = password_hash($password,
PASSWORD_DEFAULT, ['cost' => 12]);
    // آپدیت در دیتابیس
}
```

---

۱ .چرا نباید رمز عبور را به صورت متن ساده (Clear Text) ذخیره کرد؟

ذخیره رمز عبور به صورت متن ساده خطرناک است، زیرا در صورت نفوذ به دیتابیس، هکرها می‌توانند رمز کاربران را به راحتی بازیابی کرده و مورد سوءاستفاده قرار دهند.

---

۲ .چگونه امنیت رمز عبور را افزایش می‌دهد؟ bcrypt

bcryptرمز عبور را به همراه salt هش می‌کند و توابع کندی دارد، که حملات brute-force را بسیار سخت کرده و برای هر رمز عبور حتی مشابه هش متفاوت تولید می‌کند.

---

۳ .فرق هشینگ و رمزنگاری چیست؟

هشینگ یک طرفه است و نمی‌توان اطلاعات اصلی را بازیابی کرد؛ رمزنگاری دوطرفه است و با کلید می‌توان اطلاعات اصلی را بازیابی کرد.

---

۴ .تابع password_hash در PHP چه کاربردی دارد؟

این تابع برای هش‌کردن رمز عبور با الگوریتم‌هایی مثل bcrypt استفاده می‌شود تا امنیت کاربران حفظ شود.

---

۵ .چگونه می‌توان یک رمز عبور هش‌شده را اعتبارسنجی کرد؟

رمز عبور واردشده توسط کاربر را مجدداً هش می‌کنیم و با مقدار هش ذخیره‌شده در دیتابیس مقایسه می‌کنیم. در PHP با تابع password_verify امکان‌پذیر است.

---

۵ تمرین با جواب

۱ .هش کردن رمز عبور با bcrypt درPHP

```
$password = 'mySecret123';
$hash = password_hash($password, PASSWORD_BCRYPT);
// نتیجه: $hash یک رشته هش‌شده منحصربه‌فرد از رمز عبور است
```

---

۲ .اعتبارسنجی رمز عبور ورودی با مقدار هش‌شده

```
$hash =
'$2y$10$.vGA1O9wmRjrwAVXD98HNOgsNpDczlqm3Jq7KnEd1rVAGv3Fykk1
a'; // رمز ذخیره شده
if(password_verify('mySecret123', $hash)){
    echo "رمز عبور صحیح است!";
} else {
    echo "رمز عبور اشتباه است.";
}
```

---

۳ .افزودن نمایش ایمیل کاربر یا Guest در صفحه

```
<p><?= $_SESSION['user']['email'] ?? 'Guest' ?></p>
```

---

۴ .استفاده از cost سفارشی برایbcrypt

```
$options = ['cost' => 12];
$hash = password_hash('hellobcrypt', PASSWORD_BCRYPT, $options);
// هش با سختی بالاتر تولید می‌شود
```

---

۵ .ذخیره امن کاربر جدید با رمز هش‌شده در دیتابیس

```
$db->query("
    INSERT INTO users (email, password) values (:email, :password)",
    [':email' => $email, ':password' => password_hash($password,
PASSWORD_BCRYPT)]
);
// رمز عبور کاربر به صورت امن ذخیره می‌شود
```

چرا ذخیره کردن پسوردها به صورت متن ساده خطرناک است؟

پاسخ:
ذخیره کردن پسوردها به صورت متن ساده خطرناک است زیرا:

- در صورت نفوذ به دیتابیس، حمله‌کنندگان به تمام اطلاعات کاربران دسترسی پیدا می‌کنند
- کاربران معمولاً از یک پسورد برای چندین سرویس استفاده می‌کنند
- امکان سوءاستفاده از اطلاعات برای ورود به حساب‌های کاربری در سایر سایتها وجود دارد
- نقض حریم خصوصی و امنیت کاربران محسوب می‌شود

سوال ۲

تابع `password_hash` چه کاری انجام می‌دهد و پارامترهای آن چیست؟

پاسخ:
تابع `password_hash` برای هش کردن پسوردها استفاده می‌شود:

- پارامتر اول: پسورد کاربر به صورت متن ساده
- پارامتر دوم: الگوریتم هش کردن
(مثل `PASSWORD_BCRYPT` یا `PASSWORD_DEFAULT`)
- مقدار بازگشتی: یک رشته هش شده که باید در دیتابیس ذخیره شود

سوال ۳

تفاوت بین `PASSWORD_DEFAULT` و `PASSWORD_BCRYPT` چیست؟

پاسخ:

- `PASSWORD_DEFAULT`: الگوریتم پیش‌فرض PHP که ممکن است در آینده تغییر کند
- `PASSWORD_BCRYPT`: الگوریتم bcrypt به طور خاص که همیشه از این الگوریتم استفاده می‌کند

- در حال حاضر هر دو از bcrypt استفاده می‌کنند، اما `PASSWORD_BCRYPT` تضمین می‌کند که همیشه از این الگوریتم استفاده شود

سوال ۴

چگونه می‌توانیم در صفحه اصلی تشخیص دهیم کاربر لاگین کرده یا خیر؟

پاسخ:
با استفاده از شرط زیر در view:

php
```
<?= $_SESSION['user']['email'] ?? 'Guest' ?>
```

اگر کاربر لاگین کرده باشد ایمیل او نمایش داده می‌شود، در غیر این صورت "Guest" نشان داده می‌شود.

سوال ۵

چرا الگوریتم bcrypt برای هش کردن پسوردها مناسب است؟

پاسخ:
bcrypt مناسب است زیرا:

- الگوریتمی بسیار امن و مطمئن است
- cracking آن بسیار زمان‌بر و دشوار است
- به صورت صنعتی استاندارد شده است
- در برابر حملات brute-force مقاومت بالایی دارد

تمرین‌ها و پاسخ‌ها

تمرین ۱

یک تابع برای بررسی قدرت پسورد بنویسید که حداقل ۸ کاراکتر و شامل حروف بزرگ، کوچک و اعداد باشد.

پاسخ:

```php
function isPasswordStrong($password) {
    $length = strlen($password) >= 8;
    $uppercase = preg_match('/[A-Z]/', $password);
    $lowercase = preg_match('/[a-z]/', $password);
    $number = preg_match('/[0-9]/', $password);

    return $length && $uppercase && $lowercase && $numb
er;
}

// تست تابع
echo isPasswordStrong('Weak1') ? 'قوی' : 'ضعیف'; // خروجی
: ضعیف
echo isPasswordStrong('StrongPass123') ? 'قوی' : 'ضعیف';
// خروجی: قوی
```

تمرین ۲

سیستم ثبت‌نام را طوری تغییر دهید که پسوردها را با الگوریتم پیش‌فرض هش کند.

پاسخ:

```php
// در فایل store.php
```

```php
$hashedPassword = password_hash($password, PASSWORD_DEFAULT);

$db->query("INSERT INTO users (email, password) VALUES (:email, :password)", [
    ':email' => $email,
    ':password' => $hashedPassword
]);
```

تمرین ۳

یک تابع برای verify کردن پسورد کاربر هنگام لاگین بنویسید.

پاسخ:

php
```php
function verifyPassword($enteredPassword, $hashedPasswordFromDB) {
    return password_verify($enteredPassword, $hashedPasswordFromDB);
}

// مثال استفاده
$enteredPassword = 'user123';
$storedHash = '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi';

if (verifyPassword($enteredPassword, $storedHash)) {
    echo 'پسورد صحیح است';
} else {
    echo 'پسورد نادرست است';
}
```

تمرین ۴

صفحه‌ای ایجاد کنید که اطلاعات کاربر لاگین کرده را نمایش دهد یا اگر مهمان است پیام مناسب نشان دهد.

پاسخ:

php
```php
// در فایل profile.php
session_start();

$user = $_SESSION['user'] ?? null;

if ($user) {
    echo "خوش آمدید " . htmlspecialchars($user['email']);
    echo "<br>اطلاعات پروفایل شما...";
} else {
    echo "لطفاً برای مشاهده پروفایل خود وارد شوید";
    echo "<a href='login.php'>ورود به سیستم</a>";
}
```

تمرین ۵

یک سیستم ساده برای بررسی expiration session بنویسید.

پاسخ:

php
```php
// در ابتدای هر صفحه که نیاز به session دارد
session_start();

// تنظیم timeout به ۳۰ دقیقه
$timeout = 1800; // 30 دقیقه به ثانیه

if (isset($_SESSION['last_activity']) &&
```

```php
    (time() - $_SESSION['last_activity']) > $timeout) {

    // session منقضی شده
    session_unset();
    session_destroy();
    echo "session شما منقضی شده است. لطفاً مجدداً وارد شوید";
    exit;
}

// به روز رسانی زمان آخرین فعالیت
$_SESSION['last_activity'] = time();
```