

اختبار

SQL Injection

1. **الوصف:** لديك صفحة تسجيل دخول تحتوي على الحقليين التاليين: اسم المستخدم وكلمة المرور. في الشيفرة الخلفية، يتم تنفيذ الاستعلام التالي للتحقق من صحة المستخدم:

```
SELECT * FROM users WHERE username = '$username' AND password = '$password';
```

المهمة:

- اشرح لماذا هذه الشيفرة عرضة لـ SQL Injection.
 - اقترح طريقة لإصلاحها.
- لأن النص المأخوذ من المستخدم (مثل اسم المستخدم وكلمة المرور) يُدرج مباشرة في الاستعلام SQL دون أي تحقق أو معالجات. هذا يمكن أن يسمح للمهاجم بإدخال نصوص SQL خبيثة كجزء من الإدخال، مما قد يؤدي إلى اختراق قاعدة البيانات أو الوصول إلى معلومات غير مصرح بها.

لإصلاح هذه المشكلة وتقليل خطر هجوم SQL Injection:

1. استخدم الاستعلامات المُعدة (Prepared Statements) التي تسمح بتمرير قيم المعلومات بشكل آمن.
2. التحقق من البيانات المدخلة من قبل المستخدم للتأكد من أنها تتبع نمط متوقع.
3. قصر صلاحيات قاعدة البيانات بحيث يمكن فقط الوصول إلى البيانات المطلوبة لتنفيذ العملية.

XSS

1. **الوصف:** لديك صفحة تعرض تعليقات المستخدمين دون تنقية المدخلات. يمكن لأي مستخدم إدخال تعليق يحتوي على شيفرة JavaScript:

```
<div><?php echo $comment; ?></div>
```

المهمة:

- اشرح لماذا هذه الشيفرة عرضة لـ XSS.
 - اقترح طريقة لإصلاحها.
- سبب تعرض الشيفرة لـ XSS:** الشيفرة تعرض التعليقات كما هي دون أي تنقية للمدخلات. أي مستخدم يمكنه إدخال تعليق يحتوي على شيفرة JavaScript.

طريقة لإصلاح المشكلة

إزالة أو ترميز الأحرف الخاصة، مثل < و >، لمنع تنفيذ الشيفرة. في PHP، يمكن استخدام htmlspecialchars() أو strip_tags()

CSRF

1. الوصف: لديك نموذج لتغيير كلمة المرور يتم إرساله عبر طلب POST

```
<form method="POST" action="change_password.php">
  <input type="password" name="new_password">
  <input type="submit" value="Change Password">
</form>
```

المهمة:

- اشرح لماذا هذا النموذج عرضة لـ CSRF.
- اقترح طريقة لإصلاحها.

عند إرسال طلب تغيير كلمة المرور عبر POST، يمكن للمهاجم إعداد صفحة خبيثة تحتوي على نموذج مشابه يقوم بإرسال الطلب بشكل غير مباشر عند فتح الضحية للصفحة، مما يؤدي إلى تنفيذ الطلب دون علم الضحية.

استخدام رمز: CSRF (CSRF Token) إضافة رمز مميز يتم إنشاؤه لكل جلسة، ويجب التحقق منه عند تقديم النموذج.