Exercise 3 (10 points)

Consider the following piece of PHP code that implements an authentication functionality by means of a username/password form:

```
$name = $_POST['username'];
$pwdHash = password_hash($_POST['pwd'], PASSWORD_DEFAULT);
$res = mysqli_query($link,
   "SELECT * FROM users WHERE name = '$name' AND pwdHash='$pwdHash'"
);
if (!$res) { die('Query error'); }
$row = mysqli_fetch_row($res);
if (!$row) { die('Invalid user ID or password'); }
// … begin session
```

Does the code contain any vulnerability? If yes, give an example of how an attacker can exploit it. How should the code be corrected?

SOLUTION

The code works normally if the inputs are those expected, but the behavior is unexpected if the input contains SQL symbols. Consider an attacker that sends the malicious user ID “admin' -- " (with any password), which contains SQL symbols that are interpreted as code by the victim system. The “” character closes the string literal in the query string, and the “-- “ (with the trailing space) symbol makes the SQL interpreter to ignore the successive query, thus bypassing the password check. The attacker gains access to the system with the privileges of the user “admin”. This is an example of SQL injection. The following code snippet shows how to solve the vulnerabiluty by escaping inputs with the mysql_real_escape_string() PHP function.

```
$name = $_POST['username'];
$pwdHash = password_hash($_POST['pwd'], PASSWORD_DEFAULT);
$res = mysqli_query($link,
  "SELECT * FROM users
  WHERE name = 'mysql_real_escape_string($name,$link)'
    AND pwdHash='mysql_real_escape_string($pwdHash,$link)'"
);
if (!$res) { die('Query error'); }
$row = mysqli_fetch_row($res);
if (!$row) { die('Invalid user ID or password'); }
// … begin session
```