

PHP

(Security,
from <http://phpsec.org>)

Intro

- Security must be part of the design
- Consider illegitimate uses of your application.
 - A secure design is only part of the solution.
 - During development, when the code is being written, it is important to consider penggunaan yg keliru dari aplikasi anda.
 - Artinya perhatian developer bukan hanya aplikasi dapat dijalankan namun juga aman

https

- Keamanan komunikasi client server aspek sgt sangat penting
- Di tingkat protokol jaringan dikenal https (http secure)
- HTTPSecure = **HTTP + secure communication protocol**
- Secure Communication Protocol such as SSL
- SSL, Secure Security Layer, provide protocol security over internet
- Memiliki standart setting security dg SSL



This Connection is Untrusted

You have asked Firefox to connect securely to **hsbc.co.uk**, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

Get me out of here!

▼ Technical Details

hsbc.co.uk uses an invalid security certificate.

The certificate is only valid for www.hsbc.co.uk

(Error code: ssl_error_bad_cert_domain)

▶ I Understand the Risks

Filtering

- If nothing else, **FILTER ALL EXTERNAL DATA.**
 - **Data filtering** is the cornerstone of web application security in any language and on any platform.
 - By initializing your variables and filtering all data that comes from an external source
 - A **whitelist approach** is better than a blacklist approach. This means that you should consider all data invalid unless it can be proven valid.

- Make sure
 - data filtering cannot be bypassed,
 - that invalid data cannot be mistaken for valid data,
and
 - Identify the origin of data.
- Avoid dengan mendayagunakan String function,number function dan secure URL

```

$noerror = array(); //array temporary valid atau no error
                // atau semacam whitelist
$email_pattern = '/^[^@\s<&>]+@[(-a-z0-9]+\.)+[a-z]{2,}$/i';
if (preg_match($email_pattern, $_POST['email']))
{
    $noerror['email'] = $_POST['email']; //associate array sering menjadi pilihan
                                        // index dengan nama tertentu bukan angka
}

if ($_POST['numint'] == strval(intval($_POST['numint'])))
    //make sure tipe adalah integer
{
    $noerror['numint'] = $_POST['numint'];
}
if ($_POST['numflo'] == strval(floatval($_POST['numflo'])))
    //make sure tipe adalah floating
{
    $noerror['numflo'] = $_POST['numflo'];
}

```

Naming

- Dont let use common variable, esp. superglobal var
- Biasakan memberikan penamaan khusus atas variable value anda
 - Ex: `$UserNama = $_POST['name']`
 - `$_POST` → common
 - `$UserNama` → not really common

Filesystem Security

- PHP is subject to the security built into most server systems with respect to permissions on a file and directory basis
- This allows you to control which files in the filesystem may be read
- Care should be taken with any files which are world readable to ensure that they are safe for reading by all users who have access to that filesystem

- Since PHP was designed to allow user level access to the filesystem, it's entirely possible to write a PHP script that will allow you to read system files
- Make sure the path, directory or files that be allowed for user to access. Each user may be different

Cookies

- For storing data in the remote browser and
- thus tracking or identifying return users, karena pada dasarnya client/server bersifat connectionless
- Tidak terlalu direkomendasikan, karena cookies dapat di-off kan
- Cookies be recorded di client, tersimpan di browser
- Be destroyed if user close the browser

- `setcookies(name,value,expire,path,domain,http)`
 - Name, nama set cookies
 - Value, nilai cookies
 - Expire, masa hidup cookies
 - Path, path tertentu di domain di mana cookies hidup
 - Domain, nama domain tertentu
 - Secure, T/F untuk https
 - `httponly`, T/F hanya untuk protokol http

- Diberikan di header sebelum yg lain

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
  <?php
    $value = 'dww';
    setcookie("UrCookie", $value);
    setcookie("UrCookie", $value, time()+1); // expire in 1'
    setcookie("UrCookie", $value, time()+1, "/WPCourse/", "localhost");
    //with define path and domain
  ?>
```

```
</html>
```

```
<?php
  echo "Welcome, " . $_COOKIE["UrCookie"]; //just to show

?>
<h3>You are accepted to fill it..</h3>
<form name="f" action="SaveFormtoTxt.php" method="post">
  <table>
    <tr>
      <td>
        Name
      </td>
      <td>
        <input name="nama" type="text" value=<? echo $_COOKIE[UrCookie]?> class="c">
      </td>
    </tr>
  </table>
</form>
```

Welcome, dww

You are accepted to fill it..

Name

dww

Comment

send

- But will be destroyed within 1'
- Anda dapat membuat modifikasi value cookies dengan akses log in member dari database → next

- Cookie can only be used to store information that the server is given
- Easier to program.
- Require less of the server

- Untuk menghapus, kosongkan nilainya

```
//Coo.php  
$value = 'dww';  
setcookie("UrCookie", $value);  
header("location:http://localhost/WPCourse/Coo3.php");
```

```
//Coo3.php  
echo "Welcome, ".$_COOKIE["UrCookie"]; //cookies kiriman dr Coo.php  
setcookie("UrCookie", ""); //dan terhapus setelah refresh
```

Welcome, dww

Setelah refresh, value kosong

Welcome,

Login

- Login is a common way to secure some pages from unauthorized user
- A login process involves just a few components:
 - A form for submitting the login information
 - A validation routine that confirms the necessary information was submitted
 - A database query that compares the submitted information against the stored information
 - Cookies or sessions to store data that reflects a successful login

- Subsequent pages will then contain checks to confirm that the user is logged in (to limit access to that page).
- There is also, of course, a logging out process, which involves clearing out the cookies or session data representing a logged-in status.

Ex:

- Use the previous ex, save user in database

Fullname:

Email :










User_ID :

Password: Retype :

Input below sign

- Input several users

- Also automatically release temporary pass in "temp" field

			fullname	email	user_id	password	temp
<input type="checkbox"/>			Doso Sedoso	doso@uns.ac.id	bosdoso	sibos	bm9abURQ
<input type="checkbox"/>			Dono Sudono	dono@yahoo.com	doidoi	2doi321	cUJMTU9U
<input type="checkbox"/>			Jono Sujono	jono@hotmail.com	jojo	jojo123	MWNwR01Q
<input type="checkbox"/>			Mono Sumono	mono@uns.ac.id	mohmoh	hom*hom**	YWR2RVRV

- Make simple login (user_id and password)

Login

User_ID:

Password:

Login

```
<body>
<form name="f" method="post" action="Signin.php">
<h2>Login</h2>
User_ID:<br>
<input name="id" size="25" type="text" class="c"><br>
Password:<br>
<input name="p" size="25" type="password" class="c"><br>
<input name="login" value="Login" type="submit" class="c">
</form>
```

- First step, make sure the input user is the real member

- Signin.php

```
if(isset($_POST['login'])){
//first check the login data in database, simple validation
require('Ceklogin.php');
if ($c_result==5){
echo "<h2>Welcome</h2>";
echo "Hi, ".$result2[2]."<br>";
require('Logged.php');
}else{
header('Location:http://localhost/public_html/Php1/Login2.php');
}}
```

```
//Ceklogin.php
$v1 = $_REQUEST['id'];$v2=$_REQUEST['p'];
mysql_connect("localhost", "root", "");
mysql_select_db("dbphp");
$q = "SELECT * FROM `member` WHERE `user_id`='$v1' AND `password`='$v2'";
$result1 = mysql_query($q);
$result2 = mysql_fetch_row($result1);
$c_result = count($result2);
```

- And page when login success

Login

User_ID:

Password:

Welcome

Hi, jojo

Here is webpage only for user

- Or unsuccessful

Login

User_ID:

Password:

Login

User_ID:

Password:

Invalid user id or password

- Next add cookies for valid user, before send any information send cookies first
- Use `setcookies()`;

```
if(isset($_POST['login'])){\n    //first check the login data in database, simple validation\n    require('Ceklogin.php');\n    if ($c_result==5){//just simple validation\n        setcookie("user",$result2[2]);//send cookies name is user\n        setcookie("pass",$result2[3]);//another cookies name is pass\n        //if true redirect to Logged.php, only user as member can access it\n        header('Location:http://localhost/public_html/Php1/Logged.php');\n        exit();//quit script\n    }else{\n        //if false or failed redirect to Login2.php\n        header('Location:http://localhost/public_html/Php1/Login2.php');\n    }\n}
```

- Modify Logged.php, use global variable `$_COOKIE[]`, to access cookies

```
if((isset($_COOKIE['user'])) and (isset($_COOKIE['pass']))) {  
    //will be showed when cookies is found  
    echo "<h2>Welcome</h2>";  
    echo "Hi, " . $_COOKIE['user'] . "<br>";  
    echo "Here is webpage only for user";  
}else{  
    header('Location:http://localhost/public_html/Php1/Login.php');  
}
```

- Lets check the cookies, input right user

Login

User_ID:

Password:

Welcome

Hi, jojo

Here is webpage only for user

- Try to close all windows but dont close the browser, then access again Logged.php



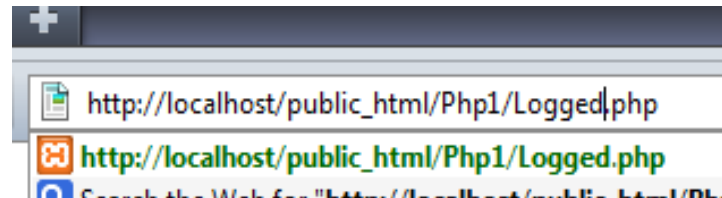
Welcome

Hi, jojo

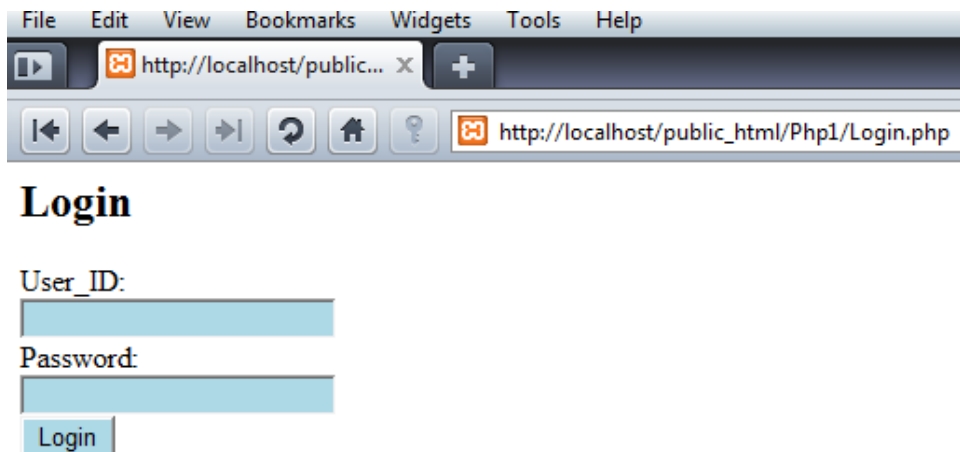
Here is webpage only for user

- the cookies still in browser, so the data still exist

- Now, close the browser then access again Logged.php



- Will be directed to Login.php, the cookies is not exist anymore



- Modify and add several page for user

```
//Logged.php
if((isset($_COOKIE['user'])) and (isset($_COOKIE['pass']))) {
    //will be showed when cookies is found
    echo "<h2>Welcome</h2>";
    echo "Hi, " . $_COOKIE['user'] . "<br>";
    echo "Here is webpage only for user<br><br>";
    echo "<a href='Logged2.php'>". "Go to the next page". "</a>". "<br>"; //add link
    echo "<a href='Logged3.php'>Go to the another page</a>"; //add another link
} else {
    header('Location:http://localhost/public_html/Php1/Login.php');
}
```

- Add Logged2.php and Logged3.php

```
//Logged2.php
if((isset($_COOKIE['user'])) and (isset($_COOKIE['pass']))) {
    //will be showed when cookies is found
    echo "<h2>Welcome</h2>";
    echo "Hi, " . $_COOKIE['user'] . "<br>";
    echo "Here is Logged2.php";
} else {
    header('Location:http://localhost/public_html/Php1/Login.php');
}

//Logged3.php
if((isset($_COOKIE['user'])) and (isset($_COOKIE['pass']))) {
    //will be showed when cookies is found
    echo "<h2>Welcome</h2>";
    echo "Hi, " . $_COOKIE['user'] . "<br>";
    echo "Here is Logged3.php";
} else {
    header('Location:http://localhost/public_html/Php1/Login.php');
}
```

- Lets check it

Welcome

Hi, jojo

Here is webpage only for user

[Go to the next page](#)

[Go to the another page](#)



Welcome

Hi, jojo

Here is Logged2.php



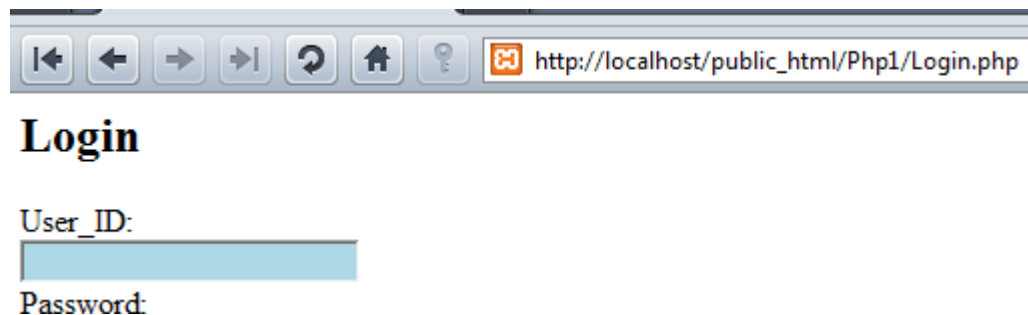
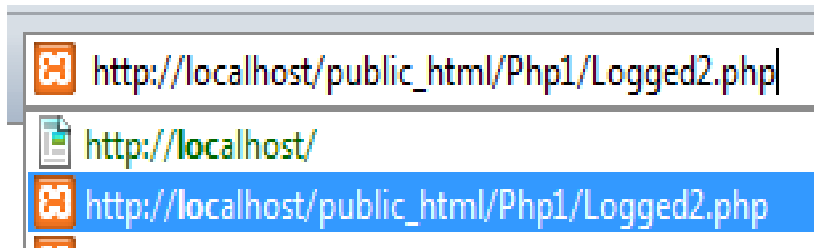
Welcome

Hi, jojo

Here is Logged3.php

- During cookies exist, page for user always validate

- Try to access it if cookies not exist anymore, will be directed to Login.php



- U can set cookies with time, eventhough browser still open the cookies will be deleted

- ```
if(isset($_POST['login'])){\n //first check the login data in database, simple validation\n require('Ceklogin.php');\n if ($c_result==5){//just simple validation\n setcookie("user",$result2[2],time()+10);//cookies life for 10 sec\n setcookie("pass",$result2[3],time()+10);//cookies life for 10 sec\n //if true redirect to Logged.php, only user as member can access it\n header('Location:http://localhost/public_html/Php1/Logged.php');\n exit();//quit script\n }else{\n //if false or failed redirect to Login2.php\n header('Location:http://localhost/public_html/Php1/Login2.php');\n }\n}
```

- U can also delete the cookies
- Make a simple sign out

```
//Signout.php
if((isset($_COOKIE['user'])) and (isset($_COOKIE['pass']))) {
 //will be showed when cookies is found
 setcookie("user","");//set no value to delete cookies
 setcookie("pass","");//set no value to delete cookies
 header('Location:http://localhost/public_html/Php1/Login.php');
} else {
 header('Location:http://localhost/public_html/Php1/Login.php');
}
```

- After sign out redirected to Login.php

- And make simple link to sign out

```
//Logged.php
if((isset($_COOKIE['user'])) and (isset($_COOKIE['pass']))) {
 //will be showed when cookies is found
 echo "Sign out
"; //link to sign out
 echo "<h2>Welcome</h2>";
 echo "Hi, " . $_COOKIE['user'] . "
";
 echo "Here is webpage only for user

";
 echo "". "Go to the next page". "". "
"; //add link
 echo "Go to the another page"; //add another link
} else {
 header('Location:http://localhost/public_html/Php1/Login.php');
}
```

- Lets check, input the right user



[Sign out](#)

## Welcome

Hi, jojo  
Here is webpage only for user

[Go to the next page](#)

[Go to the another page](#)

- After sign out will be directed to Login.php
- U can put the link sign out anywhere U want

# Session

- Another method of making data available to multiple pages of a Web site is to use sessions.
- The premise of a session is that **data is stored on the server**, not in the Web browser, and a session identifier is used to locate a particular user's record (the session data).
- This session identifier is normally **stored in the user's Web browser via a cookie**, but the sensitive data itself—like the user's ID, name, and so on—always remains on the server.

- More secure (because the data is being retained on the server).
- Allow for more data to be stored.
- Can be used without cookies, coz sometimes user turn off cookies on their browser

# Ex:

- Use prev Login.php
- Modify the handle file

```
//using session
if(isset($_POST['login'])) {
 //first check the login data in database, simple validation
 require('Ceklogin.php');
 if ($c_result==5){//just simple validation
 session_start();//start session
 $_SESSION['user']= $result2[2];//changed from setcookie
 $_SESSION['pass']= $result2[3];
 //if true redirect to Logged.php, only user as member can access it
 header('Location:http://localhost/public_html/Php1/LoggedSession.php');
 exit();//quit script
 }else{
 //if false or failed redirect to Login2.php
 header('Location:http://localhost/public_html/Php1/Login2.php');
 }
}
```



```

//LoggedSession.php
session_start();
if((isset($_SESSION['user'])) and (isset($_SESSION['pass']))) {
 //will be showed when cookies is found
 echo "Sign out
"; //link to
 echo "<h2>Welcome</h2>";
 echo "Hi, " . $_SESSION['user'] . "
";
 echo "Here is webpage only for user

";
 echo "". "Go to the next page". "". "
"; //ac
 echo "Go to the another page"; //add another
} else {
 header('Location:http://localhost/public_html/Php1/Login3.php');
}

```

```

//LoggedSession2.php
session_start();
if((isset($_SESSION['user'])) and (isset($_SESSION['pass']))) {
 //will be showed when cookies is found
 echo "<h2>Welcome</h2>";
 echo "Hi, " . $_SESSION['user'] . "
";
 echo "Here is LoggedSession2.php using SESSION not COOKIE";
} else {
 header('Location:http://localhost/public_html/Php1/Login.php');
}

```

```

//LoggedSession3.php
session_start();
if((isset($_SESSION['user'])) and (isset($_SESSION['pass']))) {
 //will be showed when cookies is found
 echo "<h2>Welcome</h2>";
 echo "Hi, " . $_SESSION['user'] . "
";
 echo "Here is LoggedSession3.php using SESSION not COOKIE";
} else {
 header('Location:http://localhost/public_html/Php1/Login.php');
}

//Signout2.php
session_start();
if((isset($_SESSION['user'])) and (isset($_SESSION['pass']))) {
 //will be showed when cookies is found
 session_unset();
 session_destroy();//delete session
 header('Location:http://localhost/public_html/Php1/Login3.php');
} else {
 header('Location:http://localhost/public_html/Php1/Login3.php');
}

```

- Lets check

## Login

User\_ID:

Password:



[Sign out](#)

## Welcome

Hi, jojo  
Here is webpage only for user

[Go to the next page](#)  
[Go to the another page](#)



## Welcome

Hi, jojo  
Here is LoggedSession2.php using SESSION not COOKIE



## Welcome

Hi, jojo  
Here is LoggedSession3.php using SESSION not COOKIE

- Session is unique "tag name" from server to user. Can be accessed by using session\_id();

```
...
• echo "Sign out
"; //lir
echo "<h2>Welcome</h2>";
echo "Hi, " . $_SESSION['user'] . ", ur session id: " . session_id() . "
";
echo "Here is webpage only for user

";
```

[Sign out](#)

**Welcome**

Hi, jojo, ur session id: b097af57a5605330f99bff725704f5c1

Here is webpage only for user

- Session ID sering disebut juga token
- Beberapa aplikasi dengan session/cookies:
  - Counter
  - Quiz online
  - Election online

# Session Hijacking

- Because important information is normally stored in a session (you should never store sensitive data in a cookie), security becomes more of an issue
- With sessions there are two things to pay attention to: **the session ID**, which is a reference point to the session data, and the **session data itself**, stored on the server
- A malicious person is far more likely to hack into a session through the session ID than the data on the server

- The session ID is the key to the session data. By default, PHP will store this in a cookie, which is preferable from a security standpoint
- If I can learn another user's session ID, I can easily trick a server into thinking that their session ID is my session ID
- Storing the **session ID in a cookie makes it somewhat harder to steal**

- One method of preventing hijacking is to **store some sort of user identifier in the session**, and then to repeatedly double-check this value
- The **HTTP\_USER\_AGENT**—a combination of the browser and operating system can be used
- This adds a layer of security in that one person could only hijack another user's session if they are **both running the exact same browser and operating system**



- Modify

```
//using session
if(isset($_POST['login'])) {
//first check the login data in database, simple validation
require('Ceklogin.php');
if ($c_result==5){//just simple validation
session_start();//start session
$_SESSION['user']= $result2[2];//changed from setcookie
$_SESSION['pass']= $result2[3];
$_SESSION['agent']= md5($_SERVER['HTTP_USER_AGENT']);//make sure the right user
//if true redirect to Logged.php, only user as member can access it
header('Location:http://localhost/public_html/Php1/LogSession.php');
exit();//quit script
}else{
//if false or failed redirect to Login2.php
header('Location:http://localhost/public_html/Php1/Login2.php');
}}
```

```

//LogSession.php
session_start();
if((isset($_SESSION['user'])) and (isset($_SESSION['pass']))) {
 if(isset($_SESSION['agent'])== md5($_SERVER['HTTP_USER_AGENT'])){//additional validation
//will be showed when cookies is found
echo "Sign out
";//link to sign out
echo "<h2>Welcome</h2>";
echo "Hi, " . $_SESSION['user'] . "
";
echo "Here is webpage only for user

";
echo "". "Go to the next page". "". "
";//add link
echo "Go to the another page";//add another link
}}else{
 header('Location:http://localhost/public_html/Php1/Login3.php');
}

```

```

//LogSession2.php
session_start();
if((isset($_SESSION['user'])) and (isset($_SESSION['pass']))) {
 if(isset($_SESSION['agent'])== md5($_SERVER['HTTP_USER_AGENT'])) {
//will be showed when cookies is found
echo "<h2>Welcome</h2>";
echo "Hi, " . $_SESSION['user'] . "
";
echo "Here is LogSession2.php using SESSION not COOKIE with user agent";
}}else{
 header('Location:http://localhost/public_html/Php1/Login3.php');
}

```

```

//LogSession3.php
session_start();
if((isset($_SESSION['user'])) and (isset($_SESSION['pass']))) {
 if(isset($_SESSION['agent'])== md5($_SERVER['HTTP_USER_AGENT'])) {
 //will be showed when cookies is found
 echo "<h2>Welcome</h2>";
 echo "Hi, " . $_SESSION['user'] . "
";
 echo "Here is LogSession3.php using SESSION not COOKIE with user agent";
 }else{
 header('Location:http://localhost/public_html/Php1/Login3.php');
 }

 //Signout2.php
 session_start();
 if((isset($_SESSION['user'])) and (isset($_SESSION['pass']))) {
 if(isset($_SESSION['agent'])== md5($_SERVER['HTTP_USER_AGENT'])) {
 //will be showed when cookies is found
 session_unset();
 session_destroy();//delete session
 header('Location:http://localhost/public_html/Php1/Login3.php');
 }else{
 header('Location:http://localhost/public_html/Php1/Login3.php');
 }
 }
}

```

- Lets check, no different in the beginning but more secure

## Login

User\_ID:

Password:



[Sign out](#)

## Welcome

Hi, jojo

Here is webpage only for user

[Go to the next page](#)

[Go to the another page](#)



## Welcome

Hi, jojo

Here is LogSession2.php using SESSION not COOKIE with user agent



## Welcome

Hi, jojo

Here is LogSession3.php using SESSION not COOKIE with user agent

# SQL Injection

- These are endeavors to insert **bad code into a site's SQL queries**
- One aim of such attacks is that they would create a syntactically invalid query, thereby revealing something about the script or database in the resulting error message
- An even bigger aspiration is that the **injection attack could alter, destroy, or expose the stored data**
- Fortunately SQL injection attacks are rather easy to prevent

- Start by **validating all data** to be used in queries (and perform typecasting, whenever possible)
- Second, use a function like **mysqli\_real\_escape\_string()** which makes data safe to use in queries.
- An alternative to using **mysqli\_real\_escape\_string()** is to use prepared statements

- The benefits of prepared statements are important: greater security and potentially better performance
- These attacks are **mainly based on exploiting the code not being written with security in mind.**
- **Never trust any kind of input**, especially that which comes from the client side, even though it comes from a select box, a hidden input field or a cookie.

- Never connect to the database as a superuser or as the database owner. Use always **customized users with very limited privileges**.
- Check if the given input **has the expected data type**. PHP has a wide range of input validating functions
- If the application waits for numerical input, consider verifying data with **is\_numeric()**
- **Quote each non numeric** user supplied value that is passed to the database with the database-specific string escape function (e.g. `mysql_real_escape_string()`, `sqlite_escape_string()`, etc.)



# Ex:

- `<?php`

```
$query = "UPDATE usertable SET pwd='$pwd'
WHERE uid='$uid';";
```

```
?>
```

- But a malicious user submits the different value
- `' or uid like'%admin%'; -- to $uid to change the admin's password, or simply sets $pwd to "hehehe', admin='yes', trusted=100 "` (with a trailing space) to gain more privileges.
- Then, the query will be twisted

- <?php

```
// $uid == ' or uid like '%admin%'; --
```

```
$query = "UPDATE usertable SET pwd='...'
WHERE uid=" or uid like '%admin%'; --";
```

```
// $pwd == "hehehe", admin='yes', trusted=100 "
```

```
$query = "UPDATE usertable SET
pwd='hehehe', admin='yes', trusted=100
WHERE...;";
```

```
?>
```

- <?php

```
$query = "SELECT * FROM products WHERE id LIKE '%$prod%'";
```

```
$result = mssql_query($query);
```

- ?>

- And sometimes malicious user submit :

```
a%' exec master..xp_cmdshell 'net user test
testpass /ADD' -- to $prod
```

- And the query become ...

- `<?php`

```
$query = "SELECT * FROM products
```

```
WHERE id LIKE '%a%'
```

```
exec master..xp_cmdshell 'net user
test testpass /ADD'--";
```

```
$result = mssql_query($query);
```

```
?>
```

- If this application were running as sa and the SERVER service is running with sufficient privileges, the attacker would now have an account with which to access this machine

# Validating Data by Type

- For the most part, the form validation used in this book thus far has been rather minimal, often just checking if a variable has any value at all
- In many situations, this really is the best you can do, easy
- Just need more time to make type validation, with built in **function PHP or make Ur own function (UDF)** depend on the aims

## Type Validation Functions

---

| FUNCTION                   | CHECKS FOR                                       |
|----------------------------|--------------------------------------------------|
| <code>is_array()</code>    | Arrays                                           |
| <code>is_bool()</code>     | Booleans (TRUE, FALSE)                           |
| <code>is_float()</code>    | Floating-point numbers                           |
| <code>is_int()</code>      | Integers                                         |
| <code>is_null()</code>     | NULLS                                            |
| <code>is_numeric()</code>  | Numeric values, even as a string<br>(e.g., '20') |
| <code>is_resource()</code> | Resources, like a database<br>connection         |
| <code>is_scalar()</code>   | Scalar (single-valued) variables                 |
| <code>is_string()</code>   | Strings                                          |