Some Basic Security Concerns in PHP

Christofer Ferm
How to protect yourself from

SQL injection
Register Globals
Session Security
Error Reporting
SQL Injection

An example of SQL Injection

Hi, this is your son’s school. We’re having some computer trouble.

Oh, dear - did he break something? In a way...

Did you really name your son Robert?); DROP TABLE Students;-- ?

Oh, yes. Little Bobby Tables, we call him.

Well, we’ve lost this year’s student records. I hope you’re happy.

And I hope you’ve learned to sanitize your database inputs.
SQL Injection

Let's brake it down

The source probably looked something like this

```php
$sql = "INSERT INTO Students (name) VALUES ('$_POST['student_name']');";

mysql_query($sql);
```

But with the input `Robert '); DROP TABLE Students; --` there will be two queries executed like this

```sql
INSERT INTO students (name) VALUES ('Robert'); DROP TABLE Students;--
```
SQL Injection

So what can the school do to avoid this?

They could do a sanity check on the input like this if they use mysql

```php
<?php
    $db = mysql_connect('localhost', 'username', 'password');
    mysql_select_db('school', $db);
    $studentName = mysql_real_escape_string($_POST['student_name'], $db);
    $queryResult = mysql_query("INSERT INTO Students (name) VALUES ('{$studentName}')");
    if ($queryResult) {
        echo 'Success.';
    } else {
        echo 'Insertion failed. Please try again.';
    }
?>
```
SQL Injection

Is there any other way do this?
Yes one example is PDO (PHP Data Objects)
It would look something like this

```php
<?php
    $db = new PDO('mysql:host=localhost;dbname=school', 'username', 'password');
    $stmt = $db->prepare('INSERT INTO Students (name) VALUES (:name)');
    try {
        $stmt->execute(array('name' => $_POST['student_name']));
        echo 'Success.';
    } catch(PDOException $e) {
        echo 'Insertion failed. Please try again.';
    }
?>
```
SQL Injection

Things to think about

Check user input

Never use root login to the database server from your php script
Register Globals

Register Globals is Available in versions older than 5.3.0
In 5.3.0 it was deprecated
In 6.0.0 it was removed

Security issues with register globals

Register Globals allows you to call a script with variables

http://www.mysite.com/auth.php?authorized=1

Here we call the script auth with the variable authorized set to 1
RegisterGlobals

Why is this a problem?
It does not necessarily need to be a problem
But it can be a security risk if a mistake is made in the code

Example

```php
<?php
if (authenticated_user()) {
    $authorized = true;
}
if ($authorized) {
    include "highly/sensitive/data.php";
}
?>
```
Register Globals

This Code becomes a problem
When register globals is on
Why?
Well if you call the script like this

http://www.mysite.com/auth.php?authorized=1
Then authorized will be true
The result from if(authenticated_user())
will not matter
Register Globals

So what is the fix?
The simple fix is pretty obvious
Initialize the variable $authorized = false;
Example with fixes

```php
<?php
$authorized = false;
if (authenticated_user()) {
    $authorized = true;
} else {

    $authorized = false;

}

if ($authorized) {
    include "/highly/sensitive/data.php";
} ?>```
RegisterGlobals

Things to think about
Always initialize variables
Consider turning Register Globals off
Sessions Security

Why do we need this?

Sessions can not breach your application's security so why bother?

That is true but it can compromise user accounts.
Sessions Security

Session information
Is stored as cookie on the client computer
Containing a session ID associated with some data on the server.
If the user has a valid session id the data associated with it will go in a global array called $_SESSIONS
Sessions Security

What are the risks?

Session stealing
Someone steals your ID
If the session is used to keep you logged in this malicious user can no enter your account using your session ID
Sessions Security

What are the risks?

If you share a web server with other users
There is yet another security issue to consider
The session data will be stored in /tmp by default (Linux machine)
This folder is readable and writable by all users on that machine.
Sessions Security

Solutions?

Session ID Stealing

re-generate id often using `session_regenerate_id()`

It will reduce the risk of ID Stealing

Session data stored in /tmp

Change the location, save data where only you have access. You can do this by `session.save_path`
Error Reporting

Error reporting is good right?
Well when developing yes
When your system is on-line no
Error Reporting

A malicious user can use Error messages to gain insight into your system

If there is a bug you want to know it and so does the hacker, he wants to exploit it and you want to fix it.
Error Reporting

You should give error messages
But keep them general
Write details to a log file that only you can access.
Error Reporting

Recommended settings for online system

Set `display_errors` to off
Set `log_errors` to on (this will log errors to file)
Set `error_log` to your log file path
Conclusion

Some suggestions

Check user input

Do not login as root to the database server from your php scripts

Do not return detailed errors write them to a log file instead

Regenerate session ID often and don't save session data at default path /tmp

Always initialize variables

Consider turning register globals off
THE END

Thank you for listening!