**PHP Security Function: strip\_tags, filter\_var, Md5 and sha1**

**Potential security threats**

They are basically two groups of people that can attack your system

* Hackers – with the intent to gain access to unauthorized data or disrupt the application
* Users – they may innocently enter wrong parameters in forms which can have negative effects on a website or web application.

The following are the kinds of attacks that we need to look out for.

**SQL Injection** – This type of attack appends harmful code to[SQL](https://www.guru99.com/sql.html)statements.

This is done using either user input forms or URLs that use variables.

The appended code comments the condition in the WHERE clause of an SQL statement. The appended code can also;

* insert a condition that will always be true
* delete data from a table
* update data in a table
* This type of attack is usually used to gain unauthorized access to an application.

**Cross-site scripting –** this type of attack inserts harmful code usually JavaScript. This is done using user input forms such as contact us and comments forms. This is done to;

* Retrieve sensitive information such as cookies data
* Redirect the user to a different URL.
* Other threats can include – PHP code injection, Shell Injection, Email Injection, Script Source Code Disclosure etc.

**PHP Application Security Best Practices**

Let’s now look at some of the PHP Security best practices that we must consider when developing our applications.

**PHP strip\_tags**

The strip\_tags functions removes HTML,[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)or PHP tags from a string.

This function is useful when we have to protect our application against attacks such as cross site scripting.

Let’s consider an application that accepts comments from users.

<?php

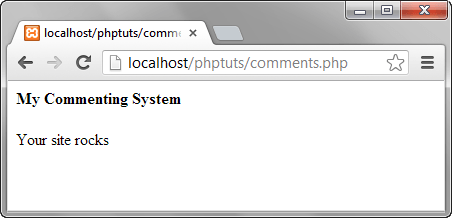
$user\_input = "Your site rocks";

echo "<h4>My Commenting System</h4>";

echo $user\_input;

?>

Assuming you have saved comments.php in the phptuts folder, browse to the URL**http://localhost/phptuts/comments.php**

[](https://www.guru99.com/images/2013/04/normal_comment.png)

Let’s assume you receive the following as the user input   <script>alert('Your site sucks!');</script>

<?php

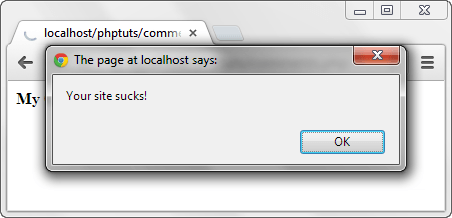
$user\_input = "<script>alert('Your site sucks!');</script>";

echo "<h4>My Commenting System</h4>";

echo $user\_input;

?>

  Browse to the URL **http://localhost/phptuts/comments.php**

[](https://www.guru99.com/images/2013/04/attacked_comment.png)

Let’s now secure our application from such attacks using strip\_tags function.

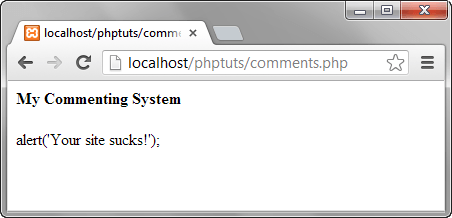
<?php

$user\_input = "<script>alert('Your site sucks!');</script>";

echo strip\_tags($user\_input);

?>

Browse to the URL**http://localhost/phptuts/comments.php**

[](https://www.guru99.com/images/2013/04/sanitized_comment.png)

**PHP filter\_var function**

The filter\_var function is used to validate and sanitize data.

Validation checks if the data is of the right type. A numeric validation check on a string returns a false result.

Sanitization is removing illegal characters from a string.

Check this link for the complete reference [filter\_var](http://www.php.net/manual/en/function.filter-var.php)

The code is for the commenting system.

It uses the filter\_var function and FILTER\_SANITIZE\_STRIPPED constant to strip tags.

<?php

$user\_input = "<script>alert('Your site sucks!');</script>";

echo filter\_var($user\_input, FILTER\_SANITIZE\_STRIPPED);

?>

**Output:**

alert('Your site sucks!');

  Mysql\_real\_escape\_string function This function is used to protect an application against SQL injection.

Let’s suppose that we have the following SQL statement for validating the user id and password.

<?php

SELECT uid,pwd,role FROM users WHERE uid = 'admin' AND password = 'pass';

?>

  A malicious user can enter the following code in the user id text box. ' OR 1 = 1 -- And 1234 in the password text box Let’s code the authentication module

<?php

$uid = "' OR 1 = 1 -- ";

$pwd = "1234";

$sql = "SELECT uid,pwd,role FROM users WHERE uid = '$uid' AND password = '$pwd';";

echo $sql;

?>

  The end result will be

SELECT uid,pwd,role FROM users WHERE uid = '' OR 1 = 1 -- ' AND password = '1234';

HERE,

* “SELECT \* FROM users WHERE user\_id = ''” tests for an empty user id
* “' OR 1 = 1 “ is a condition that will always be true
* “--" comments that part that tests for the password.

The above query will return all the users Let’s now use mysql\_real\_escape\_string function to secure our login module.

<?php

$uid = mysql\_real\_escape\_string("' OR 1 = 1 -- ");

$pwd = mysql\_real\_escape\_string("1234");

$sql = "SELECT uid,pwd,role FROM users WHERE uid = '$uid' AND password = '$pwd';";

echo $sql;

?>

  The above code will output

SELECT uid,pwd,role FROM users WHERE uid = '\' OR 1 = 1 -- ' AND password = '1234';

*Note the second single quote has been escaped for us, it will be treated as part of the user id and the password won’t be commented.*

**PHP Md5 and PHP sha1**

Md5 is the acronym for Message Digest 5 and sha1 is the acronym for Secure Hash Algorithm 1.

They are both used to encrypt strings.

Once a string has been encrypted, it is tedious to decrypt it.

Md5 and sha1 are very useful when storing passwords in the database.

The code below shows the implementation of md5 and sha1

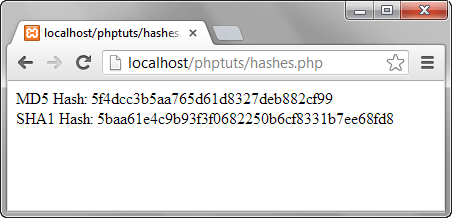
<?php

echo "MD5 Hash: " . md5("password");

echo "SHA1 Hash: " . sha1("password");

?>

  Assuming you have saved the file hashes.php in phptuts folder, browse to the URL

[](https://www.guru99.com/images/2013/04/hashes.png)

As you can see from the above hashes, if an attacker gained access to your database, they still wouldn’t know the passwords for them to login.

**Summary**

* Security refers to measures put in place to protect an application from accidental and malicious attacks.
* strip\_tags function is used to remove tags such as <script></script> from input data
* filter\_var function validates and php sanitize input data
* mysql\_real\_escape\_string is used to sanitize SQL statement. It removes malicious characters from the statements
* both MD5 and SHA1 are used to encrypt password.