ow to secure your php application

ted by Stanislav Furman on February 21, 2012

ng on web projects you should consider most common dangers such as XSS ("Cross Site ng"), SQL Injection, Session Hijacking, Spoofed Form Submissions, etc. This little guide will ou to understand basics of web security and it will help you to protect your web applications t possible attacks.

nber that security must be balanced with expense. Don't use paranoidal security approaches if ot really needed.

note that this post covers only basic security approach which is quite enough for most of web ut not enough for large, e-commerce projects. However, listed measures are fundamental for es of web applications.

1. Do not display errors on your web site.

Turn off error displaying. Detailed error can give an idea how to attack your web site. Use log files and email notifications to be informed about any errors occured on your web site. Make sure that you use something like this:

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
ini_set('display_errors', 0);
ini_set('log_errors', 1);
ini_set('error_log', 'your_path/error_log.txt');
?>
```

2. Spoofed form submissions: check where the data

comes from.

Remember that incoming data can come not only from a form located on your website, but it also car come from a form created by another person outside your web server or web site. Look at this example:

```
<form action="myform.php" method="POST">
<select name="make">
    <option value="toyota">Toyota</option>
    <option value="honda">Honda</option>
    <option value="chevy">Chevrolet</option>
</select>
<input type="submit" value="Send data" />
</form>
```

Here you can see the HTML code of a form located on your web server. You suppose that only one of three car makes can come to your script. However, a hacker can create a form on his server or even his localhost:

```
<form action="http://yoursite.com/myform.php" method="POST">
<input name="make" type="text" />
<input type="submit" value="Send data" />
</form>
```

In a such way a hacker can send any data to your script. That's why it is very important to check where the data comes from. You can do it by using a random token stored in the session. Take a look at the following example.

PHP code:

```
<?php
// generate token
$_SESSION['token'] = md5(uniqid(mt_rand(), true));
?>
```

HTML code:

```
<form method="post" action="index.php">
<input name="token" type="hidden" value="<?=$_SESSION['token']?>" />
        ...
</form>
```

PHP code (after form submission):

```
<?php
if($_SESSION['token'] != $_POST['token'] || !isset($_SESSION['token'])) {
        die('Hacker?');
}
?>
```

3. Check your input data.

Check and filter all data that comes from input fields. Make sure that you check data types, data length and cut HTML tags from the incoming requests.

Trim data if needed. If you save first name and last name to the database, you don't need to save spaces before/after (users can add useless spaces accidentally).

```
<?php $username = trim($_POST['username']); ?>
```

Remove HTML tags if they shouldn't be there:

```
<?php $username = strip_tags(trim($_POST['username'])); ?>
```

Check the size of incoming data and cut it if needed:

```
<?php $language= substr($_POST['lang'], 0, 2); // Leave only first two
characters?>
```

Validate incoming data and make sure that data contains allowed values only:

```
<?php $language = (in_array($_POST['lang'], array('en', 'fr'))) ?
$_POST['lang']:'en'; ?>
```

Forcibly convert the data to proper type:

```
<?php $age = intval($_POST['age']); ?>
```

4. Do not insert unescaped strings to the database.

Use mysql_real_escape_string() function instead of addslashes() in order to secure your application from SQL injections:

```
<?php
$username = mysql_real_escape_string($_POST['username']); ?>
```

5. Do not store passwords in plain text!

Always use hashing functions such as md5 or sha1 to encrypt passwords.

```
<?php $password = md5($_POST['password']); ?>
```

In theory, if a hacker gains access to the password hash, he could use a "dictionary" of hashed words to find a hash that matches the user's one. Therefore, always use "salting" in order to better protect passwords saved in the database.

```
<?php
$salt = md5("a salt string that nobody knows");
$encrypted_password = md5( $password.$salt );
?>
```

Salted hashes can not be cracked by using a "dictionary" of hashed words because nobody knows with what string you have "salted" the password.

6. Check files that users upload on your site.

It is absolutely necessary to check all files that users can upload using your web site. Do not allow files that can be executed by a hacker (*.php, *.exe, *.js, etc). You could filter uploaded files by file extensions using the following example:

```
<?php
$allowedFileExtensions = array(
    'txt',
    'rar',
    'zip'
);
list($filename, $extension) = explode(".",$_FILES['image']);
if(!in_array($extension, $allowedFileExtensions)) {
    die('Sorry, this file can not be uploaded.');
}
?>
```

You could also check MIME-types to filter files:

```
<?php
$allowedMimeTypes = array(
    'image/jpeg'
        'image/png',
    'image/gif'
);
$picture = $_FILES['image'];
if(!in_array($picture['type'], $allowedMimeTypes)) {
    die('Sorry, this file can not be uploaded.');
}
?>
```

7. Protect files that people shouldn't see.

There are often some files (usually executable scripts) such as PHP libraries that you include to your application code. Usually these files have extensions like .php and can be executed by a **$_GET** request. Of course, it is not always dangerous, but it is always undesirable. You could use a protected folder to store such files. To protect the folder just place in it an **.htaccess** file with the following instruction:

```
deny from all
```

Now, if anybody will try to access any file from this folder, the server will return the HTTP error code 403.

8. Session hijacking and session fixation.

Session Hijacking is one of the most common types of attack. Therefore, it is very important to prevent this danger. In theory your session identifier can be captured and used by a hacker. Using your valid session identifier a hacker can send HTTP requests to your server so the server will think that it is you.

First of all, make sure that you have properly setup your php.ini file. Use session.cookie-lifetime" and gc-maxlifetime to specify expiration time of a session cookie.

You can also control session lifetime in your PHP code. This method is very recommended. The idea is very simple - store the most recent activity time in the session and destroy session if new activity happened after limited time.

```
<?php
// Expire session after 30 minutes
if (isset($_SESSION['last_activity']) && (time() - $_SESSION['last_activity'] >
1800)) {
    session_destroy();
    session_unset();
}
$_SESSION['last_activity'] = time();
```

```
?>
```

For more protection you can regenerate session id, for example, every 15 minutes:

```
<?php
if (!isset($_SESSION['creation_time'])) {
    $_SESSION['creation_time'] = time();
} else if (time() - $_SESSION['creation_time'] > 900) {
    session_regenerate_id(true);
    $_SESSION['CREATED'] = time();
}
?>
```

Another method to prevent session hijacking is a permanent control of user's HTTP headers such as IP address and User-Agent. Usually, IP address doesn't change within a visit on your web site and the User-Agent always stays the same. That means that within a session we can simply store all this information and permanently check it in the code:

```
<?php
session_start();
if ( isset($_SESSION['HTTP_USER_AGENT']) && isset($_SESSION['REMOTE_ADDR']) )
{
        if(     $_SESSION['HTTP_USER_AGENT'] != md5($_SERVER['HTTP_USER_AGENT'])
                || $_SESSION['REMOTE_ADDR'] != md5($_SERVER['REMOTE_ADDR'])) {
                header("location login.php");
        }
} else {
    $_SESSION['HTTP_USER_AGENT'] = md5($_SERVER['HTTP_USER_AGENT']);
        $_SESSION['REMOTE_ADDR'] = md5($_SERVER['REMOTE_ADDR']);
}
?>
```

9. Use HTTPS to protect data exchanging.

If you have a payment system on your website, it is strictly recommended to use HTTPS connection.

10. Advanced security measures.

There is no perfection in the world. As well as the perfect safety doesn't exist. Where high level safety is very important, I recommend to use more security measures that involves server administration skills. Consult the following article to secure your PHP application on server's level : 25 PHP Security Best Practices For Sys Admins.

Related posts:

- Spam with animated email subject
- Russian hackers steal 1.2 billion user credentials. Is this true?
- How to use optional parameters in URI path in Yii framework
- Securing a directory with 777 or 775 permissions
- Shared web hosting Vs VPS hosting Vs dedicated servers

php security tips tricks good practices how-tos

Comments

Alabi says:

May 5, 2014 at 06:11 am

> what could be happening I implemented the process described in "Spoofed form submissions: check where the data comes from". But a file on my server online is till been spoofed

Flag as SPAM | Permalink

Stan Furman says:

May 5, 2014 at 01:11 pm

> Hi Alabi,
>
> I did not understand your question. Could you re-phrase it please?

Flag as SPAM | Permalink

Alabi says:

May 5, 2014 at 02:26 pm

@Stan Furman
I have a form on index.php file and I added this line of code into it
$_SESSION["token"] = md5(uniqid(mt_rand(), true));.....
<form action="processor.php" method="post">
<input type="hidden" name="token" value="<?php echo $_SESSION["token"]; ?>" />.....

inside processor.php I have this lines of code
if (isset($_SESSION['token']) && ($_SESSION['token']==$_POST['token'])){
unset($_SESSION['token']);
mail('email@domain.com');
}

And I am still getting mail at email@domain.com from processor.php after I commented the the form html code from index.php
So are secured is the process you explained in 2

Flag as SPAM | Permalink

Leave your comment

Fields with * are required.

Name *

Email

Comment *

Verification Code

Get a new code

Submit comment

* When you submit a comment, you agree with Terms and Conditions of Use.

Site search

Type keywords here

Go!

Quick Poll

Do you have your own blog?

- Yes, I do
- No, not yet
- Will start it soon

Vote

Recent Posts

Find domain names in just a few seconds

Microsoft launches MS-DOS Mobile

Spam with animated email subject

Russian hackers steal 1.2 billion user credentials. Is this true?

How to use optional parameters in URI path in Yii framework

Top 5 articles

Concatenating NULL and blank fields in MySQL

How to interview a programmer. Thoughts about hiring process.

How to secure your php application

PHP application performance tuning

Securing a directory with 777 or 775 permissions

Article categories

PHP

MySQL

CSS

Security

Technology

SEO

Tags

how-tos google technology web design programming humour mysql tricks news business tips php security fun good practices