Securing web applications from all sorts of forged attacking attempts is the ultimate duty of a web developer. You should build your web apps protective enough to have no security issues or loopholes, thus eradicating the possibility of any malicious attack. In most cases, the developers must take the responsibility and put in every possible effort to identify the vulnerabilities and propose solutions, if any to address the issues prevailing in the apps.

PHP is a popular language for web development. It is so popular that In some cases companies do run few bounty programs in which they invite different security experts to analyze their application from the core and suggest critical PHP security best practices for it.

Table of Content

6. Cross site request forgery XSRF/CSRF
7. Session Hijacking
8. Hide Files from the Browser
9. Securely Upload Files
10. Use SSL Certificates For HTTPS
11. Deploy PHP Apps on Clouds
12. Document Root Setup
13. Log all Errors and Hide in Production
14. Whitelist Public IP for Mysql
15. Wrapping Up!

PHP is the most criticized scripting language when it comes to security. A major chunk of developers and QA experts think PHP has no robust techniques to secure applications. The verdict has some ground too because PHP is the oldest and widely used language for web app development. But for a long time since PHP 5.6, we haven't seen any major updates regarding security and hence the language faces some security issues.

Security Issues in PHP CMS

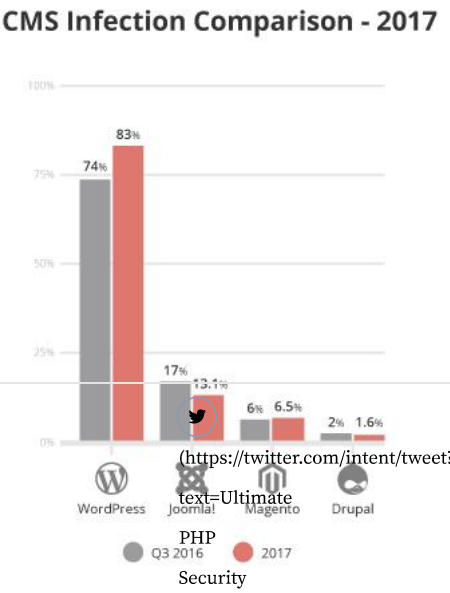
Popular CMS like WordPress, Joomla, Magento, and Drupal are built in PHP and according to Sucuri, most of the vulnerabilities in PHP CMS came to light during the year 2017:

WordPress security issues rose from 74% in 2016 Q3 to 83% in 2017.

Joomla security issues have dropped from 17% in 2016 Q3 to 13.1% in 2017.

Magento security issues rose marginally from 6% in Q3 2016 to 6.5% in 2017.

Drupal security issues dropped slightly from 2% in Q3 2016 to 1.6% in 2017.



The current situation is not good enough, but thanks to open source contributors, who are trying hard to overcome the problems and we have seen some drastic changes in PHP of late. PHP 7.x was launched last year with various updates and fixes. The best thing about PHP 7.x relates to the security upgradations which truly revamped the security protocol of the language.

What This PHP Security Tutorial Contains?

I've been working on PHP security and performance issues for a very long time, being highly active in the PHP community asking top developers about the tips and tricks they are using in their live projects. Therefore, the main aim of this PHP security tutorial is to make you aware about the best practices for security in PHP web applications. I will be defining the the following problems and mentioning possible solutions for them.

- Update PHP Regularly
- Cross site scripting (XSS)
- SQL Injection Attacks
- Cross site request forgery XSRF/CSRF
- Session Hijacking
- Hide Files from the Browser
- Securely Upload Files
- Use SSL Certificates For HTTPs
- Deploy PHP Apps on Clouds

Note: please do not consider it as a complete cheat sheet. There must be better ways and more unique solutions developers would be applying on the their applications to make it perfectly secured.

Update PHP Regularly

Right now, the most stable and latest version of PHP available is PHP 7.2.8. I recommend that you must update your PHP application to this new one. If you are still using PHP 5.6 then you will be having a lot of deprecations while upgrading PHP apps. You will also need to update your code and change some functional logics like password hashing etc. There are also some tools available to check the deprecation of your code and help you in migrating those. I have listed some tools below:

1. PHP 7 Compatibility Checker (https://github.com/sstalle/php7cc)
2. PHP 7 MAR (https://github.com/Alexia/php7mar)
3. Phan (https://github.com/etsy/phan)

If you are using PHPStorm then you can use PHP 7 Compatibility Inspection, that will show you which code will cause you issues.

Read More: PHP 7.2 Hosting on Cloudways (https://www.cloudways.com/blog/php-7-2-hosting-on-cloudways/)

Cross-site scripting (XSS)

Cross site scripting is a type of malicious web attack in which an external script is injected into the website's code or output. The attacker can send infected code to the end user while browser can not identify it as a trusted script. This attack occurs mostly on the places where user has the ability to input and submit data. The attack can access cookies, sessions and other sensitive information about the browser. Let's look at the example of a GET request which is sending some data through URL:

```
URL: http://example.com/search.php?search=<script>alert('test')</script>

$search = $_GET['search'] ?? null;

echo 'Search results for '.$search;
```

You can figure out this attack by using **htmlspecialchars**. Also by using ENT_QUOTES, you can escape single and double quotes.

```
$search = htmlspecialchars($search, ENT_QUOTES, 'UTF-8');

echo 'Search results for '.$search;
```

Meanwhile, XSS attacks can also execute via attributes, encoded URI schemes and code encoding.

Read More: Prevent XSS in Laravel (https://www.cloudways.com/blog/prevent-xss-exploits-using-laravel-validation-and-sanitization/)

SQL Injection Attacks

The SQL injection is the most common attack in PHP scripting. A single query can compromise the whole application. In SQL injection attack, the attacker tries to alter the data you are passing via queries. Suppose you are directly processing user data in SQL queries, and suddenly, an anonymous attacker secretly uses different characters to bypass it. See the below-mentioned SQL query:

```
$sql = "SELECT * FROM users WHERE username = '" . $username . "';
```

The $username can contain altered data which can damage the database including deleting the whole database in the blink of an eye. So, what's the solution? PDO. I recommend that you always use prepared statements. PDO helps you in securing SQL queries.

Let's look at another example in which GET data is sent through URL: http://example.com/get-user.php?id=1 OR id=2;

```
$id = $_GET['id'] ?? null;
```

The connection between the database and application is made with the below statement:

```
$dbh = new PDO('mysql:dbname=testdb;host=127.0.0.1', 'dbusername', 'dbpassword');
```

You can select username based on the above ID but wait! Here SQL code 'GET data' gets injected in your query. Be careful to avoid such coding and use prepared statements instead:

```
$sql = "SELECT username, email FROM users WHERE id = ".$id." ;

foreach ($dbh->query($sql) as $row) {

   printf ("%s (%s)\n", $row['username'], $row['email']);

}
```

Now you can avoid the above SQL injection possibility by using

```
$sql = "SELECT username, email FROM users WHERE id = :id";

$sth = $dbh->prepare($sql, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);

$sth->execute([':id' => $id]);

$users = $sth->fetchAll();
```

Also, a good practice is to use ORM like doctrine or eloquent, as there is the least possibility of injecting SQL queries in them.

Read More: Protect PHP Website From SQL Injection (https://www.cloudways.com/blog/protect-php-website-sql-injection/)

Cross site request forgery XSRF/CSRF

The CSRF attack is quite different to XSS attacks. In CSRF attack, the end user can perform unwanted actions on the authenticated websites and can transfer malicious commands to the site to execute any undesirable action. CSRF can't read the request data and mostly targets the state changing request by sending any link or altered data in HTML tags. It can force the user to perform state changing requests like transferring funds, changing their email addresses etc. Let's see this URL in which GET requests is sending money to another account:

```
GET http://bank.com/transfer.do?acct=TIM&amount=100 HTTP/1.1
```

Now if someone wants to exploit the web application he/she will change the URL with name and amount like this

```
http://bank.com/transfer.do?acct=Sandy&amount=100000
```

Now this URL can be sent via email in any file, Image etc and the attacker might ask you to download the file or click on the image. And as soon as you do that, you instantly end up with sending huge amount of money you never know about.

Session Hijacking

Session hijacking is a particular type of malicious web attack in which the attacker secretly steals the session ID of the user. That session ID is sent to the server where the associated $_SESSION array validates its storage in the stack and grants access to the application. Session hijacking is possible through an XSS attack or when someone gains access to the folder on a server where the session data is stored.

To prevent session hijacking always bind sessions to your IP address to:

```
$IP = getenv ( "REMOTE_ADDR" );
```

Watch for it when working on localhost as it doesn't provide you the exact IP but :::1 or :::127 type values. You should invalidate (unset cookie, unset session storage, remove traces) session quickly whenever a violation occurs and should always try not to expose IDs under any circumstances.

for cookies, the best practice is to never use serialize data stored in a cookie. Hackers can manipulate cookies easily, resulting in adding variables to your scope. Safely delete the cookies like this:

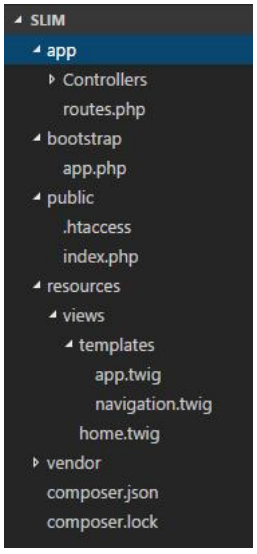
```
setcookie ($name, "", 1);

setcookie ($name, false);

unset($_COOKIE[$name]);
```

The first line of the code ensures that cookie expires in browser, the second line depicts the standard way of removing a cookie (thus you can't store false in a cookie). The third line removes the cookie from your script.

Read more: Redis Server As A PHP Session Handler (https://www.cloudways.com/blog/setup-redis-as-session-handler-php/)

Hide Files from the Browser

If you have used micro-frameworks of PHP, then you must have seen the specific directory structure which ensures the placement of files properly. Frameworks allow to have different files like controllers, models, configuration file(.yaml) etc in that directory, but most of the time browser doesn't process all the files, yet they are available to see in the browser. To resolve this issue, you must not place your files in the root directory but in a public folder so that they are not accessible all the time in browser. Look at the directory structure of the Slim framework (https://www.cloudways.com/blog/simple-rest-api-with-slim-micro-framework/) below:



Securely Upload Files

File uploading is a necessary part of any user data processing application. But remember, at some points, files are also used for XSS attacks as I have already explained above in the article. Returning to the basics, always use the POST request in the form and declare the property **enctype="multipart/form-data"** in <form> tag. Then validate the file type using finfo class like this:

```
$finfo = new finfo(FILEINFO_MIME_TYPE);
$fileContents = file_get_contents($_FILES['some_name']['tmp_name']);

$mimeType = $finfo->buffer($fileContents);
```

Developers can create their own custom and ultra-secure file validation rules, but some frameworks like Laravel, Symfony and codeigniter already have pre-defined methods to validate file types.

Let's look at another example. The HTML of form should be like this:

```
<form method="post" enctype="multipart/form-data" action="upload.php">

   File: <input type="file" name="pictures[]" multiple="true">

   <input type="submit">

</form>
```

And upload.php contains the following code:

```
foreach ($_FILES['pictures']['error'] as $key => $error) {

   if ($error == UPLOAD_ERR_OK) {

       $tmpName = $_FILES['pictures']['tmp_name'][$key];

       $name = basename($_FILES['pictures']['name'][$key]);

       move_uploaded_file($tmpName, "/var/www/project/uploads/$name");

   }

}
```

Properly declaring the **UPLOAD_ERR** and **basename()** may prevent directory traversal attacks, but few other validations – like file size, file rename and store uploaded files in private location – are also required to strengthen the security of the applications.

Read more: Image and File Upload in PHP (https://www.cloudways.com/blog/the-basics-of-file-upload-in-php/)

Use SSL Certificates For HTTPS

All the modern browsers like Google Chrome, Opera, Firefox and others, recommend to use HTTPS protocol for web applications. HTTPs provides a secured and encrypted accessing channel for untrusted sites. You must include HTTPS by installing SSL certificate into your website. It also strengthens your web applications against XSS attacks and prevents the hackers to read transported data using codes. Cloudways provides free SSL certificates on one-click which are valid for 90 days and you can easily revoke or renew them with a click. You can follow the guides for setting-up SSL Certificates in your PHP (https://www.cloudways.com/blog/add-ssl-certificates-to-custom-php-sites/), WordPress, Magento and Drupal applications.

Deploy PHP Apps on Clouds

Hosting is the final and paramount step for any web application, as you always create the project on local PHP servers (https://www.cloudways.com/blog/best-php-servers/) and deploy them on live servers which offer either shared, cloud or dedicated hosting. I always recommend to use cloud hosting like DigitalOcean, Linode, AWS (https://www.cloudways.com/en/amazon-cloud-hosting.php). They are fast, safe and secure for any kind of website and application. They always provide secured layer to prevent DDOS, Brute force and phishing attacks which are highly detrimental for web applications.

You Might Also Like: Pitfalls of Laravel Shared Hosting for Your Projects (https://www.cloudways.com/blog/stay-away-from-laravel-shared-hosting/)

To deploy PHP applications (https://www.cloudways.com/blog/deploy-php-application/) on cloud servers, you must have good Linux skills to create powerful web stacks like LAMP or LEMP, which often costs you time and budget for Linux professionals. Instead, Cloudways managed PHP and MySQL hosting (https://www.cloudways.com/en/php-cloud-hosting.php) platform provides you the easy way to deploy servers with Thunderstack within few clicks on the above-mentioned cloud providers. The Thunderstack helps your PHP application to be completely secured from various malicious attacks and guarantees optimized performance.

Read more: Enhanced Cloudways Staging Environment Is Now Available for All Users (https://www.cloudways.com/blog/announcing-enhanced-staging/)

Document Root Setup

The document root for PHP applications on any server must be set to **var/www/**html so that users can access your website via the browser. But in some cases, when you are developing APIs with frameworks like Laravel, Symfony, and Slim, you need to update the webroot to `**/public** folder.

/public serves the output of application similar to simple PHP website with index.php file. The purpose to set the webroot to **var/www/html/public** is to hide the sensitive files like .htaccess and .env which contain the environment variables and credentials of database, mail, payment APIs.

Also, frameworks like Laravel, Symfony recommend to not move all of your files to root folder, instead creating a nice directory structure to save related files like view, models and controllers is a more reasonable approach.

Log all Errors and Hide in Production

Once you have developed the website and deployed on live server. The first thing you must do is disable the display of errors, because hackers might get same valuable information from the errors. Set this parameter in your php.ini file:

```
display_errors=Off
```

Now, after making display off, log PHP errors (https://www.cloudways.com/blog/php-error-logging/) to a specific file for future needs:

```
log_errors=On

error_log=/var/log/httpd/php_scripts_error.log
```

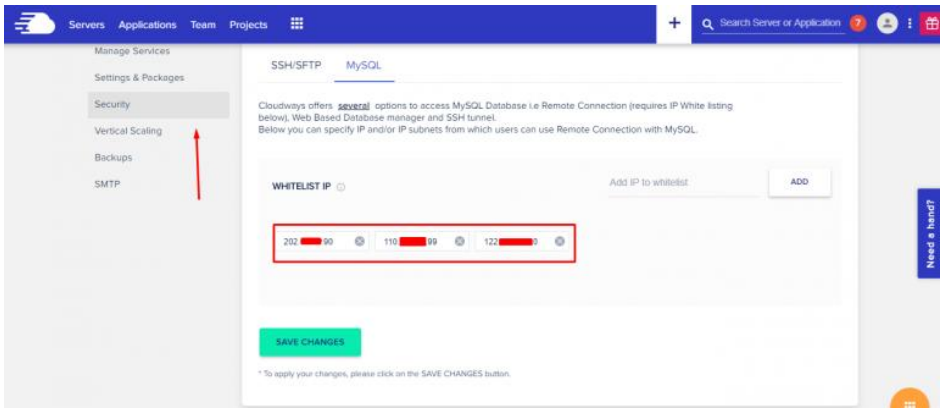
Obviously, you can change file name as you want.

You Might Also Like: Simplify Laravel Error Logging with PHP Rollbar Integration (https://www.cloudways.com/blog/laravel-error-logging-rollbar/)

Whitelist Public IP for Mysql

When working with PHP apps you frequently need to setup mysql database in internal scripts as well as in mysql clients. Most clients are used to set up MySQL remote connection (https://www.cloudways.com/blog/connect-to-remote-mysql-database/) which needs the IP address or other hostname provide by hosting server to create connection.

The public IP for the remote Mysql connection must be whitelisted in your hosting server so that an anonymous user can not get access to the database. For instance, on Cloudways you can whitelist IP as:



Now you can connect SQLyog or Mysql workbench to work remotely with database.

Wrapping Up!

Well, The PHP security best practices is a very vast topic. Developers from around the world tend to develop different use cases to secure web apps. While many companies run different bounty programs to find out security loopholes and vulnerabilities in their applications and thus reward those security experts who point out critical loopholes in the applications. In this article, I have covered basic PHP security issues, to help you understand how to secure your PHP projects from different malicious attacks. I'll also write about few more PHP security tips and tricks in the future as well. Till then you can contribute your thoughts and security practices in the comments section below.

Shahroze Nawaz (https://www.cloudways.com/blog/author/shahroze-nawaz/)

Shahroze is a PHP Community Manager at Cloudways - A Managed PHP Hosting (https://www.cloudways.com/en/php-cloud-hosting.php) Platform. Besides his work life, he loves movies and travelling. You can email him at shahroze.nawaz@cloudways.com

THERE'S MORE TO READ.

(HTTPS://WWW.CLOUDWAYS.COM/BLOG/PHP-DEVELOPMENT-COMPANIES/)

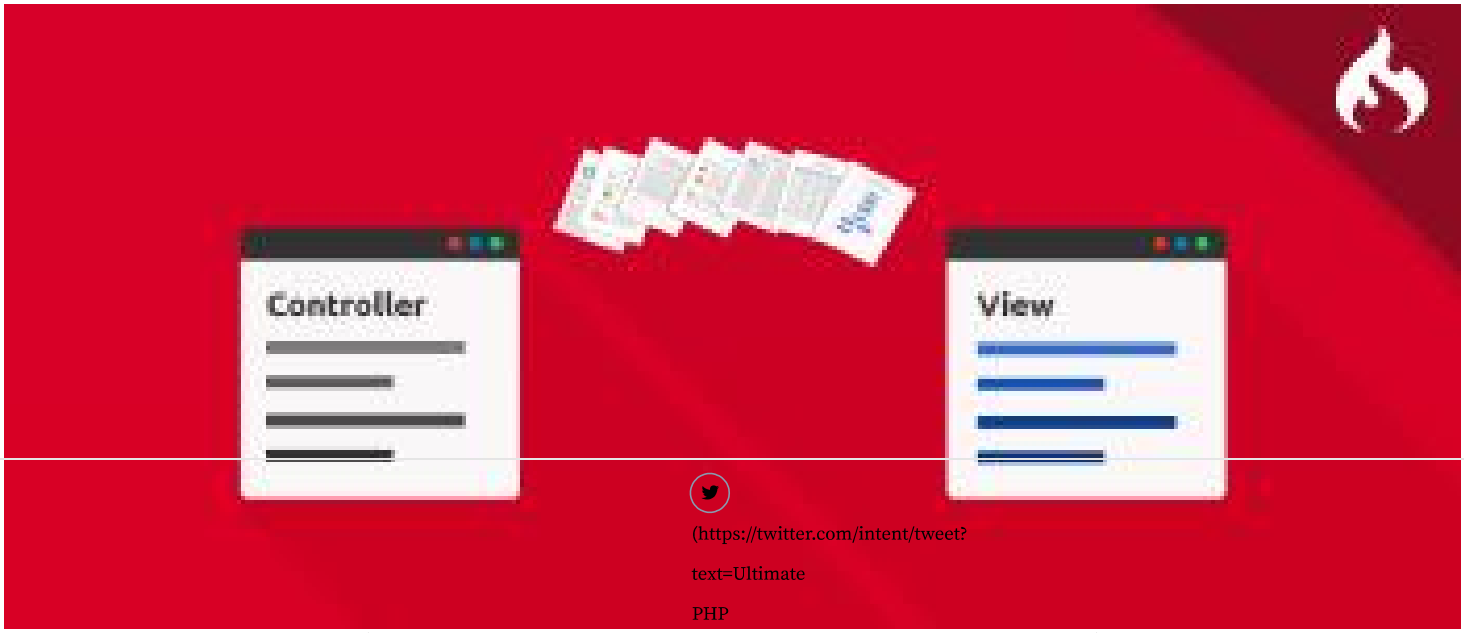
(https://twitter.com/intent/tweet?text=Ultimate PHP Security Best Practices,&url=https://www.cloudways.com/blog/php-security/&via=cloudways)

PHP

30 Best PHP Development Companies in 2019 (https://www.cloudways.com/blog/php-development-companies/)

(HTTPS://WWW.CLOUDWAYS.COM/BLOG/WOOCOMMERCE-3-7-FEATURES/)

What's new in WooCommerce 3.7: Performance Tweaks and More! (https://www.cloudways.com/blog/woocommerce-3-7-features/)



(HTTPS://WWW.CLOUDWAYS.COM/BLOG/HOW-TO-PASS-DATA-IN-CODEIGNITER/)

CodeIgniter

How to Pass Data From Controller to View in... (https://www.cloudways.com/blog/how-to-pass-data-in-codeigniter/)

(https://twitter.com/intent/tweet?text=Ultimate PHP Security Best Practices,&url=https://www.cloudways.com/blog/php-security/&via=cloudways)

6 Comments Cloudways Login

Recommend Tweet Share Sort by Best

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS

Name

Ecomsolver Private Limited • 5 months ago
Woww great tips
Expecting more such informative pieces on your blog.
I will follow these practice.
3 • Reply • Share ›

pardeep kumar ➚ Ecomsolver Private Limited • 5 months ago • edited
@Ecomsolver Private Limited thanks for follow this article, I'm glad you like it.
• Reply • Share ›

Ashish Shah • 6 months ago
If you want to develop secure PHP based website than this above points are very useful to implement in development.
• Reply • Share ›

Shahroze Nawaz ➚ Ashish Shah • 5 months ago
Thanks **@Ashish Shah** for the shoutout. I'll add more tips soon :)
• Reply • Share ›

Visualmodo WordPress Themes • a year ago
Awesome article, thanks for share!
• Reply • Share ›

Shahroze Nawaz ➚ Visualmodo WordPress Themes • a year ago
Thanks for the good words I'll update with more tips in future :)
1 • Reply • Share ›

ALSO ON CLOUDWAYS

Server & Application Monitoring Now Simplified
1 comment • 4 months ago
James Joseph Finn — You missed telling us about the "New Relic" tab. Is New Relic included in CloudWays?

"We Do Love Cloudways", Says Ryan Sullivan of WP Site Care
1 comment • 7 months ago
Sunyyan Junaid — Spiderman: Into the Spiderverse is indeed worth a watch! D

What Is Ecommerce & How Can You Make Money with It in 2019
1 comment • 6 months ago
AnnaFis — A really good article. The ecommerce industry is a great and very big topic.

Is Your Store Safe From Magento Killer?
1 comment • a month ago
Leo — wow, thanks for sharing! :)

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy

PRODUCT & SOLUTION

WordPress Hosting (https://www.cloudways.com/en/wordpress-cloud-hosting.php)

Magento Hosting (https://www.cloudways.com/en/magento-managed-cloud-hosting.php)

PHP Cloud Hosting (https://www.cloudways.com/en/php-cloud-hosting.php)

Laravel Hosting (https://www.cloudways.com/en/laravel-hosting.php)

Drupal Hosting (https://www.cloudways.com/en/drupal-cloud-hosting.php)

(https://twitter.com/intent/tweet?text=Ultimate PHP Security Best Practices,&url=https://www.cloudways.com/blog/php-security/&via=cloudways)

Joomla Hosting (https://www.cloudways.com/en/joomla-cloud.php)

PrestaShop Hosting (https://www.cloudways.com/en/prestashop-hosting.php)

WooCommerce Hosting (https://www.cloudways.com/en/hosting-woocommerce.php)

Cloudways Platform (https://platform.cloudways.com/signup)

Cloudways API (https://developers.cloudways.com/)

Breeze – Free WordPress Cache (https://www.cloudways.com/en/free-wordpress-cache-plugin-breeze.php)

Add-ons (https://www.cloudways.com/en/addons.php)

CloudwaysCDN (https://www.cloudways.com/en/cdn-services.php)

CloudwaysBot (https://www.cloudways.com/en/cloudwaysbot.php)

COMPANY

SUPPORT

QUICK LINKS

Follow Us On

(https://www.facebook.com/cloudways) (https://twitter.com/cloudways) (https://www.linkedin.com/company/cloudways) (https://www.cloudways.com/blog/updates/)

(https://www.cloudways.com/en/)

52 Springvale, Pope Pius XII Street Mosta MST2653, Malta

(https://twitter.com/intent/tweet?text=Ultimate PHP Security Best Practices,&url=https://www.cloudways.com/blog/php-security/&via=cloudways)