How to secure database passwords in PHP?

When a PHP application makes a database connection it of course generally needs to pass a login and password. If I'm using a single, minimum-permission login for my application, then the PHP needs to know that login and password somewhere. What is the best way to secure that password? It seems like just writing it in the PHP code isn't a good idea.

php database security

edited Jan 13 '11 at 7:53
AviD
9,583 4 45 69

asked Sep 18 '08 at 23:27
user18359
1,468 4 12 9

1 To be totally secure, you'll need to set up an ssl connection, otherwise anyone on your network can still sniff the password you type. – Charles Ma Sep 18 '08 at 23:31

1 Do you mean the user passwords or the database password used in the connection string? – Ozgur Ozcitak Sep 18 '08 at 23:31

3 Database password used in the connection string. Thanks! – user18359 Sep 19 '08 at 0:29

15 Answers

Several people misread this as a question about how to **store** passwords in a database. That is wrong. It is about how to store the password that lets you get **to** the database.

The usual solution is to move the password out of source-code into a configuration file. Then leave administration and securing that configuration file up to your system administrators. That way developers do not need to know anything about the production passwords, and there is no record of the password in your source-control.

edited Nov 7 '11 at 19:51
Farray
4,459 1 19 31

answered Sep 18 '08 at 23:32
user11318
6,197 2 18 25

5 Thanks. If I understand this correctly, the php file will then have an include to the config file, allowing it to use the password. e.g. I create a file called 'app1_db_cfg.php' that stores the login, pword, & db name. Then my application.php page includes 'app1_db_cfg.php' and I'm in business! – user18359 Sep 19 '08 at 0:40

18 I agree that the config needs to be properly protected. However knowing how to do that is the business of system administrators, not developers. I disagree on the value of strong encryption in this case. If you can't protect your config file, what makes you think you can protect your keys? – user11318 Sep 21 '08 at 20:04

6 I prefer using a database account that is only allowed to acces the database from the web server. And then I don't bother encrypting the configuration, I just store it outside the web root. – gnud Apr 25 '09 at 8:01

6 I use an apache environment variable to set the path so that even the path to the file is unknown in the source code. This also nicely allows having a different password for development and production based on what Apache settings are on the serverm – geedew Mar 12 '14 at 13:07

6 Please keep in mind that even files stored outside of the web accessible directory must be read by the script that uses them. If someone includes that file, then dumps the data from the file, they will see the password. – Rick Mac Gillis Nov 8 '14 at 19:33

If you're hosting on someone else's server and don't have access outside your webroot, you can always put your password and/or database connection in a file and then lock the file using a .htaccess:

```
<files mypasswdfile>
order allow,deny
deny from all
</files>
```

answered Aug 3 '09 at 18:28

kellen
2,925 2 12 8

3 Thanks, that was exactly what I was looking for. – David Gladfelter Aug 11 '09 at 14:48

Useful, if it's truly secure, though it seems like a shell login would still have access. – Kzqai Apr 14 '10 at 16:10

20 Definitely, but if someone has shell access, your entire account has been compromised anyway. – kellen Apr 16 '10 at 18:59

There is no real solution for this. Anyone can use dump functions in file, upload the file to production server and execute it. Yayy you got all the secrets of production server ! – Ankit Feb 8 at 17:07

Store them in a file outside web root.

answered Sep 18 '08 at 23:32

da5id
5,558 5 30 47

21 And also, as mentioned elsewhere, outside of source control. – Frank Farmer May 26 '09 at 20:46

3 we would be able to include it? e.g. in PHP can we then do `include('../otherDirectory/configfile.conf')` ? – mtk Jan 5 '13 at 17:23

For extremely secure systems we encrypt the database password in a configuration file (which itself is secured by the system administrator). On application/server startup the application then prompts the system administrator for the decryption key. The database password is then read from the config file, decrypted, and stored in memory for future use. Still not 100% secure since it is stored in memory decrypted, but you have to call it 'secure enough' at some point!

answered Sep 19 '08 at 13:33

pdavis
2,702 1 20 27

50 what if the admin dies ? – Radu Murzea May 3 '13 at 18:47

11 @RaduMurzea that's ridiculous. When have you heard of Sys Admins dying? They're like McDonalds, they just appear/disappear out of nowhere! – AlanChavez Jan 23 '15 at 18:43

6 @Radu Murzea Just have 2 or more admins, then you have parity like a raid array. Chances of more than one drive failing at a time is much lower. – element11 Sep 24 '15 at 15:23

what about when the servers restart? What about the time it takes to wake the admin up to get them to type the password in..etc.etc. lol – John Hunt Jan 18 at 11:52

The most secure way is to not have the information specified in your PHP code at all.

If you're using Apache that means to set the connection details in your httpd.conf or virtual hosts file file. If you do that you can call mysql_connect() with no parameters, which means PHP will never ever output your information.

This is how you specify these values in those files:

```
php_value mysql.default.user      myusername
php_value mysql.default.password  mypassword
php_value mysql.default.host      server
```

Then you open your mysql connection like this:

```
<?php
$db = mysqli_connect();
```

Or like this:

```
<?php
$db = mysqli_connect(ini_get("mysql.default.user"),
                     ini_get("mysql.default.password"),
```

```
ini_get("mysql.default.host"));
```

edited Nov 21 '13 at 14:42

Fred -ii-
63.6k 9 35 69

answered Apr 16 '12 at 15:13

Lars Nyström
1,511 9 13

1 Please check the proper values of ini_get('default values') php.net/manual/en/class.mysqli.php – Val Jun 15 '12 at 13:08

Could one use this in their .htaccess file? – user1193509 Aug 18 '12 at 17:49

@user1193509 Yes. – untill Nov 15 '15 at 8:43

yes, but any user (or a hacker abusing badly written php script) can read the password via `ini_get()` . – Marki555 Mar 21 at 17:50

This solution is general, in that it is useful for both open and closed source applications.

1. Create an O/S user for your application. See http://en.wikipedia.org/wiki/Principle_of_least_privilege
2. Create a (non-session) O/S environment variable for that user, with the password
3. Run the application as that user

Advantages:

1. You won't check your passwords into source control by accident, because you can't
2. You won't accidentally screw up file permissions. Well, you might, but it won't affect this.
3. Survives reboot
4. Can only be read by root or that user. Root can read all your files and encryption keys anyways.
5. If you use encryption, how are you storing the key securely?
6. Works x-platform
7. You can clear the envvar value after you read it. Otherwise be careful if calling other processes, as they may see the envvar.

This method is suggested by Heroku, who are very successful.

edited Jan 22 '15 at 6:09

answered Dec 23 '13 at 21:48

Neil McGuigan
21.2k 5 47 84

Your choices are kind of limited as as you say you need the password to access the database. One general approach is to store the username and password in a seperate configuration file rather than the main script. Then be sure to store that outside the main web tree. That was if there is a web configuration problem that leaves your php files being simply displayed as text rather than being executed you haven't exposed the password.

Other than that you are on the right lines with minimal access for the account being used. Add to that

- Don't use the combination of username/password for anything else
- Configure the database server to only accept connections from the web host for that user (localhost is even better if the DB is on the same machine) That way even if the credentials are exposed they are no use to anyone unless they have other access to the machine.
- Obfuscate the password (even ROT13 will do) it won't put up much defense if some does get access to the file, but at least it will prevent casual viewing of it.

Peter

answered Sep 18 '08 at 23:36

Vagnerr
1,812 3 19 33

---

if it is possible to create the database connection in the same file where the credentials are stored. Inline the credentials in the connect statement.

```
mysql_connect("localhost", "me", "mypass");
```

Otherwise it is best to unset the credentials after the connect statement, because credentials that are not in memory, can't be read from memory ;)

```
include("/outside-webroot/db_settings.php");
mysql_connect("localhost", $db_user, $db_pass);
unset ($db_user, $db_pass);
```

edited Sep 20 '08 at 20:58

answered Sep 19 '08 at 19:25

Bob Fanger
13.4k 5 39 48

5 If someone has access to the memory, you're screwed anyway. This is pointless fake-security. Outside the webroot (or at the least protected by a .htaccess if you don't have access above your webroot) is the only safe option. – uliwitness Sep 23 '12 at 13:48

Put the database password in a file, make it read-only to the user serving the files.

Unless you have some means of only allowing the php server process to access the database, this is pretty much all you can do.

answered Sep 18 '08 at 23:32

Chris
2,878 12 15

If you're talking about the database password, as opposed to the password coming from a browser, the standard practice seems to be to put the database password in a PHP config file on the server.

You just need to be sure that the php file containing the password has appropriate permissions on it. I.e. it should be readable only by the web server and by your user account.

answered Sep 18 '08 at 23:35

Jason Wadsworth
709 8 18

Unfortunately the PHP config file can be read by phpinfo() and if someone happens to leave some test script behind a lucky attacker would be able to read the password. It's probably best to leave the connection password in a file outside the web server root instead. Then the only way to access it is either with a shell or by executing arbitrary code, but in that scenario all security is lost anyway. – MarioVilas Apr 27 '14 at 18:51

Best way is to not store the password at all!
For instance, if you're on a Windows system, and connecting to SQL Server, you can use Integrated Authentication to connect to the database without a password, using the current process's identity.

If you do need to connect with a password, first **encrypt** it, using strong encryption (e.g. using AES-256, and then protect the encryption key, or using asymmetric encryption and have the OS protect the cert), and then store it in a configuration file (outside of the web directory) with **strong ACLs**.

answered Sep 20 '08 at 21:17

AviD
9,583 4 45 69

3 No point in encrypting the password *again*. Someone who could get at the unencrypted password can also get at whatever passphrase is needed to decrypt the password. However, using ACLs & .htaccess is a good idea. – uliwitness Sep 23 '12 at 13:46

2 @uliwitness I think you may have misunderstood - what do you mean by "encrypt *again*"? It's just the one encryption. And you don't want to be using passphrases (intended for human use) to encrypt it, rather strong key management, e.g. protected by the OS, in such a way that simply accessing the file system will not grant access to the key. – AviD Sep 23 '12 at 15:29

1 Encryption is not magic - instead of protecting the AES key with ACLs you could just store the password there. There is no difference between accessing the AES key or the decrypted password, encryption in this context is just snake oil. – MarioVilas Apr 27 '14 at 18:47

@MarioVilas whaat? If the password is encrypted, with the encryption key protected by the OS, how is there no difference? Encryption is not magic - it just compacts all the secrecy into the smaller encryption key.

Hardly snakeoil, in this context it is just *moving* all that secrecy into the OS. – AviD Apr 28 '14 at 6:25

2 @AviD how come the OS can protect the key but not the data itself? Answer: it can protect both, so encryption doesn't really help. It'd be different if only the data was stored and the encryption key was derived, for example, from a password that had to be *typed* by a user. – MarioVilas Sep 22 '14 at 10:45

I think the OP means the database password.

Unless someone gains access to your server via FTP or SSH (in which case you're already buggered), I wouldn't worry about storing passwords in plaintext in PHP files. Most PHP applications I've seen do it that way, for example phpbb.

answered Sep 18 '08 at 23:31

Ryan Bigg
80k 17 168 209

If you are using PostgreSQL, then it looks in `~/.pgpass` for passwords automatically. See the manual for more information.

answered Sep 18 '08 at 23:52

Jim
48.2k 9 73 87

An additional trick is to use a PHP separate configuration file that looks like that :

```
<?php exit() ?>

[...]

Plain text data including password
```

This does not prevent you from setting access rules properly. But in the case your web site is hacked, a "require" or an "include" will just exit the script at the first line so it's even harder to get the data.

Nevertheless, do not ever let configuration files in a directory that can be accessed through the web. You should have a "Web" folder containing your controler code, css, pictures and js. That's all. Anything else goes in offline folders.

answered Sep 19 '08 at 13:08

e-satis
227k 80 224 279

but then how does the php script read the credentials stored in the file? – Christopher Mahan Jan 22 '09 at 8:40

1 You use fopen(), like for a regular text file. – e-satis Jan 23 '09 at 14:56

1 @e-satis ok it will prevent hacker to do `require` / `include` but how to prevent to do `fopen` ? – dmnc Oct 23 '13 at 12:26

"this does not prevent you from setting access rules properly" – e-satis Oct 23 '13 at 12:36

Just putting it into a config file somewhere is the way it's usually done. Just make sure you:

1. disallow database access from any servers outside your network,
2. take care not to accidentally show the password to users (in an error message, or through PHP files accidentally being served as HTML, etcetera.)

edited Mar 10 '14 at 3:14

Raptor
24.6k 21 112 204

answered Sep 18 '08 at 23:35

Marijn
925 1 6 11

protected by Bill the Lizard Sep 10 '10 at 11:13

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site.

Would you like to answer one of these unanswered questions instead?