PHP Session Security

What are some guidelines for maintaining responsible session security with PHP? There's information all over the web and it's about time it all landed in one place!

security php

edited May 4 '12 at 9:39
mattytommo
32.5k 9 52 89

asked Aug 2 '08 at 2:41
saint_groceon
2,784 4 16 23

locked by Bill the Lizard Jul 18 '12 at 17:21

This question exists because it has historical significance, but **it is not considered a good, on-topic question for this site**, so please do not use it as evidence that you can ask similar questions here. This question and its answers are frozen and cannot be changed. More info: help center.

13 Answers

There are a couple of things to do in order to keep your session secure:

1. Use SSL when authenticating users or performing sensitive operations.
2. Regenerate the session id whenever the security level changes (such as logging in). You can even regenerate the session id every request if you wish.
3. Have sessions time out
4. Don't use register globals
5. Store authentication details on the server. That is, don't send details such as username in the cookie.
6. Check the `$_SERVER['HTTP_USER_AGENT']`. This adds a small barrier to session hijacking. You can also check the IP address. But this causes problems for users that have changing IP address due to load balancing on multiple internet connections etc (which is the case in our environment here).
7. Lock down access to the sessions on the file system or use custom session handling
8. For sensitive operations consider requiring logged in users to provide their authenication details again

edited Jun 6 '12 at 17:37
user656925

answered Aug 11 '08 at 2:38
grom
8,752 10 49 61

15 Using SSL only for some operations is not enough, unless you have separate sessions for encrypted and unencrypted traffic. If you use single session over HTTPS and HTTP, attacker will steal it on first non-HTTPS request. – pomeL Nov 16 '08 at 14:41

6 -1 the user agent is trivial to spoof. What you are describing wastes code and is not a security system. – rook Apr 23 '10 at 16:32

24 @The Rook, it may be a trivial barrier (the attacker can capture a victim's user-agent using their own site) and relies on security through obscurity but it is still one extra barrier. If the User-Agent HTTP was to change during the session use, it would be extremely suspicious and most likely an attack. I never said you can use it alone. If you combine it with the other techniques you have a much more secure site. – grom Apr 27 '10 at 23:42

5 @grom I think its like putting a piece of scotch tape across your door and saying it will prevent people from breaking in. – rook Apr 27 '10 at 23:59

8 If you're checking the user agent, you'll block all requests from IE8 users when they toggle compatibility mode. See the fun I had tracking down this problem in my own code: serverfault.com/questions/200018/http-302-problem-on-ie7. I'm taking the user agent check out, because it's such a trivial thing to spoof, as others have said. – bestattendance Nov 9 '10 at 20:18

One guideline is to call session_regenerate_id every time a session's security level changes. This helps prevent session hijacking.

answered Aug 2 '08 at 2:43

saint_groceon
2,784 4 16 23

My two (or more) cents:

- Trust no one
- Filter input, escape output (cookie, session data are your input too)
- Avoid XSS (keep your HTML well formed, take a look at PHPTAL or HTMLPurifier)
- Defense in depth
- Do not expose data

There is a tiny but good book on this topic: Essential PHP Security by Chris Shiflett.

On the home page of the book you will find some interesting code examples and sample chapters.

You may use technique mentioned above (IP & UserAgent), described here: How to avoid identity theft

edited Apr 6 '10 at 16:25

answered Apr 6 '10 at 16:05

takeshin
18.8k 12 76 127

+1 for XSS-prevention. Without that it's impossible to protect against CSRF, and thus somebody can "ride" the session without even getting the session ID. – porneL Oct 27 '11 at 14:34

I think one of the major problems (which is being addressed in PHP 6) is register_globals. Right now one of the standard methods used to avoid `register_globals` is to use the `$_REQUEST`, `$_GET` or `$_POST` arrays.

The "correct" way to do it (as of 5.2, although it's a little buggy there, but stable as of 6, which is coming soon) is through filters.

So instead of:

```
$username = $_POST["username"];
```

you would do:

```
$username = filter_input(INPUT_POST, 'username', FILTER_SANITIZE_STRING);
```

or even just:

```
$username = filter_input(INPUT_POST, 'username');
```

edited May 4 '12 at 9:41

mattytommo
32.5k 9 52 89

answered Aug 2 '08 at 2:55

cmcculloh
12.9k 22 70 109

2 This has no relation to the question at all. – The Pixel Developer Aug 12 '09 at 4:02

5 Really? Then why in the accepted answer do they mention not to use register globals? Wouldn't, as far as most run-of-the-mill developers are concerned, register globals and form variable handling fall under the

umbrella of "sessions" even if it isn't technically part of the "session" object? – cmcculloh Aug 18 '09 at 18:06

-1 This does not answer the question. – Tomas Aug 17 '10 at 14:54

9 I agree, this does not *fully* answer the question, but it is definitely PART of the answer to the question. Again, this fleshes out a bullet point in the accepted answer, "Don't use register globals". This tells what to do instead. – cmcculloh Aug 18 '10 at 2:48

This session fixation paper has very good pointers where attack may come. See also session fixation page at Wikipedia.

answered Mar 5 '09 at 22:33

raspi
2,956 15 35

Using IP address isn't really the best idea in my experience. For example; my office has two IP addresses that get used depending on load and we constantly run into issues using IP addresses.

Instead, I've opted for storing the sessions in a separate database for the domains on my servers. This way no one on the file system has access to that session info. This was really helpful with phpBB before 3.0 (they've since fixed this) but it's still a good idea I think.

answered Aug 6 '08 at 20:44

Eric Lamb
820 3 13 26

This is pretty trivial and obvious, but be sure to session_destroy after every use. This can be difficult to implement if the user does not log out explicitly, so a timer can be set to do this.

Here is a good tutorial on setTimer() and clearTimer().

edited Aug 2 '08 at 3:24

answered Aug 2 '08 at 3:16

helloandre
6,253 3 31 50

The main problem with PHP sessions and security (besides session hijacking) comes with what environment you are in. By default PHP stores the session data in a file in the OS's temp directory. Without any special thought or planning this is a world readable directory so all of your session information is public to anyone with access to the server.

As for maintaining sessions over multiple servers. At that point it would be better to switch PHP to user handled sessions where it calls your provided functions to CRUD (create, read, update, delete) the session data. At that point you could store the session information in a database or memcache like solution so that all application servers have access to the data.

Storing your own sessions may also be advantageous if you are on a shared server because it will let you store it in the database which you often times have more control over then the filesystem.

answered Aug 3 '08 at 13:14

John Downey
7,552 3 25 29

I set my sessions up like this-

on the log in page:

```
$_SESSION['fingerprint'] = md5($_SERVER['HTTP_USER_AGENT'] . PHRASE .
$_SERVER['REMOTE_ADDR']);
```

(phrase defined on a config page)

then on the header that is throughout the rest of the site:

```
session_start();
if ($_SESSION['fingerprint'] != md5($_SERVER['HTTP_USER_AGENT'] . PHRASE .
$_SERVER['REMOTE_ADDR'])) {
    session_destroy();
    header('Location: http://website login page/');
    exit();
}
```

edited Jul 19 '11 at 22:00

answered Jul 19 '11 at 21:40

Chad
31 2

php.ini

```
session.cookie_httponly = 1
change session name from default PHPSESSID
```

eq Apache add header:

```
X-XSS-Protection    1
```

answered Oct 13 '11 at 2:40

Userpassword
1,382 1 15 29

Can you elaborate? – domino Nov 5 '11 at 18:57

httpd.conf -> <FilesMatch "\.(php|phtml|aspx|htm|html)$">Header set X-XSS-Protection "1"</FilesMatch> – Userpassword May 4 '12 at 13:52

Be aware that `X-XSS-Protection` isn't really useful at all. In fact, the protecting algorithm itself could actually be exploited, making it worse than before. – Pacerier Jul 13 '12 at 10:31

I would check both IP and User Agent to see if they change

```
if ($_SESSION['user_agent'] != $_SERVER['HTTP_USER_AGENT']
    || $_SESSION['user_ip'] != $_SERVER['REMOTE_ADDR'])
{
    //Something fishy is going on here?
}
```

answered Aug 4 '08 at 21:38

Teifion
23.8k 51 120 163

5 IP can legitimately change if user is behind load-balanced proxy farm. – porneL Nov 16 '08 at 14:42

2 And user_agent can change every time a user upgrades their browser. – scotts Aug 11 '09 at 19:03

3 @scotts I agree with the IP part but for the browser upgrade, you would set the session when they login so I don't see how they would upgrade there browser without creating a new session once they login again. – JasonDavis Sep 7 '09 at 11:36

I believe the user_agent can also change when toggling between compatibly mode in IE8. It's also very easy to fake. – Scott Herbert Jan 23 '11 at 11:25

Yep but what about users that had static IP eq GSM and is changed every half hour. So, stored IP in Session + host name, WHEN IP != REMOTE_ADDR check host and compare hostanmes eq. 12.12.12.holand.nl-> when is holand.nl == true. But some host had IP based hostname Then need compare mask 88.99.XX.XX – Userpassword May 4 '12 at 13:49

If you you use session_set_save_handler() you can set your own session handler. For example you could store your sessions in the database. Refer to the php.net comments for examples of a database session handler.

DB sessions are also good if you have multiple servers otherwise if you are using file based sessions you would need to make sure that each webserver had access to the same filesystem to read/write the sessions.

answered Aug 9 '08 at 3:28

ejunker
3,974 6 29 37

You need to be sure the session data are safe. By looking at your php.ini or using phpinfo() you can find you session settings. *session.save*path_ tells you where they are saved.

Check the permission of the folder and of its parents. It shouldn't be public (/tmp) or be accessible by other websites on your shared server.

Assuming you still want to use php session, You can set php to use an other folder by changing *session.save*path_ or save the data in the database by changing *session.save*handler_ .

You might be able to set *session.save*path_ in your php.ini (some providers allow it) or for apache

+ mod*php, in a .htaccess file in your site root folder:* `php value session.savepath "/home/example.com/html/session"` . *You can also set it at run time with* _session*savepath()* .

Check Chris Shiflett's tutorial or Zend*Session*SaveHandler_DbTable to set and alternative session handler.

answered Aug 18 '08 at 17:12

Dinoboff
1,224 1 14 20