Last updated: Fri, 26 Feb 2010

Using Register Globals

Warning

This feature has been ***DEPRECATED*** as of PHP 5.3.0 and ***REMOVED*** as of PHP 6.0.0. Relying on this feature is highly discouraged.

Perhaps the most controversial change in PHP is when the default value for the PHP directive register_globals went from ON to OFF in PHP » 4.2.0. Reliance on this directive was quite common and many people didn't even know it existed and assumed it's just how PHP works. This page will explain how one can write insecure code with this directive but keep in mind that the directive itself isn't insecure but rather it's the misuse of it.

When on, register_globals will inject your scripts with all sorts of variables, like request variables from HTML forms. This coupled with the fact that PHP doesn't require variable initialization means writing insecure code is that much easier. It was a difficult decision, but the PHP community decided to disable this directive by default. When on, people use variables yet really don't know for sure where they come from and can only assume. Internal variables that are defined in the script itself get mixed up with request data sent by users and disabling register_globals changes this. Let's demonstrate with an example misuse of register_globals:

Example #1 Example misuse with register_globals = on

```
<?php
// define $authorized = true only if user is authenticated
if (authenticated_user()) {
   $authorized = true;
}

// Because we didn't first initialize $authorized as false, this might be
// defined through register_globals, like from GET auth.php?authorized=1
// So, anyone can be seen as authenticated!
if ($authorized) {
   include "/highly/sensitive/data.php";
}
?>
```

When register_globals = on, our logic above may be compromised. When off, $authorized can't be set via request so it'll be fine, although it really is generally a good programming practice to initialize variables first. For example, in our example above we might have first done *$authorized = false*. Doing this first means our above code would work with register_globals on or off as users by default would be unauthorized.

Another example is that of sessions. When register_globals = on, we could also use $username in our example below but again you must realize that $username could also come from other means, such as GET (through the URL).

Example #2 Example use of sessions with register_globals on or off

```
<?php
// We wouldn't know where $username came from but do know $_SESSION is
// for session data
if (isset($_SESSION['username'])) {

   echo "Hello <b>{$_SESSION['username']}</b>";

} else {

   echo "Hello <b>Guest</b><br />";
   echo "Would you like to login?";
```

```
}
?>
```

It's even possible to take preventative measures to warn when forging is being attempted. If you know ahead of time exactly where a variable should be coming from, you can check to see if the submitted data is coming from an inappropriate kind of submission. While it doesn't guarantee that data has not been forged, it does require an attacker to guess the right kind of forging. If you don't care where the request data comes from, you can use $_REQUEST as it contains a mix of GET, POST and COOKIE data. See also the manual section on using variables from external sources.

Example #3 Detecting simple variable poisoning

```
<?php
if (isset($_COOKIE['MAGIC_COOKIE'])) {

   // MAGIC_COOKIE comes from a cookie.
   // Be sure to validate the cookie data!

} elseif (isset($_GET['MAGIC_COOKIE']) || isset($_POST['MAGIC_COOKIE'])) {

  mail("admin@example.com", "Possible breakin attempt", $_SERVER['REMOTE_ADDR']);
  echo "Security violation, admin has been alerted.";
  exit;

} else {

  // MAGIC_COOKIE isn't set through this REQUEST

}
?>
```

Of course, simply turning off register_globals does not mean your code is secure. For every piece of data that is submitted, it should also be checked in other ways. Always validate your user data and initialize your variables! To check for uninitialized variables you may turn up error_reporting() to show **E_NOTICE** level errors.

For information about emulating register_globals being On or Off, see this FAQ.

Note: **Superglobals: availability note**
Superglobal arrays such as $_GET, $_POST, and $_SERVER, etc. are available as of PHP 4.1.0. For more information, read the manual section on superglobals

User Contributed Notes
Using Register Globals

Andrew dot GuertinNO at SPAMuvm dot edu
19-Jun-2009 07:35

```
It's not mentioned anywhere, but it seems register globals also affects file uploads.

When a file is uploaded (with in this case, <input type="file" name="coconut">), the following
variables appear:

["_FILES"]=>
array(1) {
["coconut"]=>
array(5) {
  ["name"]=>
  string(14) "mozicon128.png"
  ["type"]=>
  string(9) "image/png"
  ["tmp_name"]=>
  string(14) "/tmp/phpWgtRBx"
  ["error"]=>
  int(0)
  ["size"]=>
  int(15113)
}
}

When register globals is turned on, the following variables also appear:
["coconut_name"]=>
```

```
string(14) "mozicon128.png"
["coconut_type"]=>
string(9) "image/png"
["coconut"]=>
string(14) "/tmp/phpWgtRBx"
["coconut_size"]=>
int(15113)
```

lester burlap
23-Mar-2009 05:00

```
It would make this whole issue a lot less confusing for less-experienced PHP programmers if you just
explained:

- $myVariable no longer works by default
- $_GET['myVariable'] works just fine

I'm embarrassed to say it's taken me six months since my ISP upgraded to PHP5 figure this out.  I've
completely rewritten scripts to stop using GET variables altogether.

I'm dumb.
```

claude dot pache at gmail dot com
15-Jan-2009 01:52

```
Beware that all the solutions given in the comments below for emulating register_global being off
are bogus, because they can destroy predefined variables you should not unset. For example, suppose
that you have

<?php $_GET['_COOKIE'] == 'foo'; ?>

Then the simplistic solutions of the previous comments let you lose all the cookies registered in
the superglobal "$_COOKIE"! (Note that in this situation, even with register_global set to "on", PHP
is smart enough to not mess predefined variables such as  $_COOKIE.)

A proper solution for emulating register_global being off is given in the FAQ, as stated in the
documentation above.
```

subarea AT webfire DOT biz
19-Nov-2008 11:38

```
your webspace provider has register-globals activated by standard and you don't have a chance to
turn it off? no problem anymore, here is your solution...

<?php
// ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
// this is just a workaround to kill all through register globals imported vars!
// ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
// place this script after session_start() to be sure you unregister_globals('_SESSION');
// that's all, now all through "register_globals" assigned vars are deleted from scope.
// ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

if (ini_get('register_globals') == 1)
{
if (is_array($_REQUEST)) foreach(array_keys($_REQUEST) as $var_to_kill) unset($$var_to_kill);
if (is_array($_SESSION)) foreach(array_keys($_SESSION) as $var_to_kill) unset($$var_to_kill);
if (is_array($_SERVER))  foreach(array_keys($_SERVER)  as $var_to_kill) unset($$var_to_kill);
                                    unset($var_to_kill);
}
?>

hope you like it ;)

greetz subarea
```

georg_gruber at yahoo dot com
27-Oct-2008 09:21

```
BEWARE of using register_globals = On, it's not only bad karma but highly dangerous.

Consider the following coding:

<?php
// assume $_SESSION['user'] = array('Hello', 'World');
// assume session_start() was called somewhere before.

  print('<pre>Contents of array $_SESSION[\'user\']');
  print_r($_SESSION['user']);
  print('<hr>Contents of array $user (PHP SETUP register_globals = On)');
  print_r($user);
```

```
  print('</pre>');
?>

If you manipulate $user you'll manipulate $_SESSION['user'] as well with PHP SETUP register_globals
= On.

So please avoid it at any cost, no serious programmer would ever want to have register_globals = On.
```

bohwaz
31-Aug-2008 09:39

```php
<?php

// Unregister_globals: unsets all global variables set from a superglobal array
// --------------------
// This is useful if you don't know the configuration of PHP on the server the application
// will be run
// Place this in the first lines of all of your scripts
// Don't forget that the register_global of $_SESSION is done after session_start() so after
// each session_start() put a unregister_globals('_SESSION');

function unregister_globals()
{
  if (!ini_get('register_globals'))
  {
    return false;
  }

  foreach (func_get_args() as $name)
  {
    foreach ($GLOBALS[$name] as $key=>$value)
    {
      if (isset($GLOBALS[$key]))
        unset($GLOBALS[$key]);
    }
  }
}

unregister_globals('_POST', '_GET', '_COOKIE', '_REQUEST', '_SERVER', '_ENV', '_FILES');

?>
```

moore at hs-furtwangen dot de
14-Jul-2008 08:19

I had a look at the post from Dice, in which he suggested the function unregister_globals(). It didn't seem to work - only tested php 4.4.8 and 5.2.1 - so I made some tweaking to get it running. (I had to use $GLOBALS due to the fact that $$name won't work with superglobals).

```php
<?php
//Undo register_globals
function unregister_globals() {
  if (ini_get('register_globals')) {
    $array = array('_REQUEST', '_FILES');
    foreach ($array as $value) {
      if(isset($GLOBALS[$value])){
        foreach ($GLOBALS[$value] as $key => $var) {
          if (isset($GLOBALS[$key]) && $var === $GLOBALS[$key]) {
            //echo 'found '.$key.' = '.$var.' in $'.$value."\n";
            unset($GLOBALS[$key]);
          }
        }
      }
    }
  }
}
?>
```

The echo was for debuging, thought it might come in handy.

fab dot mariotti at [google]gmail dot com
16-Apr-2008 07:59

For my application I defined two functions:
wit_set_gv('space','key','value')
wit_get_gv('space','key')
Forgive the "wit_" prefix but the gv stays for Global Variable.

Maybe I should start with a simple version:
wit_set_gv('key','value')
wit_get_gv('key')

This way you would set or get a global/session value.
The register_globals (on or off), session state and/or
superglobal variables will be handled by these functions.

I did add a 'space' item because I wanted to have control
on what goes to/comes from where. As an example if I call:
wit_get_gv('WIT_CONF','URL')
I know that I have to check for a global variable named
WIT_CONF which also gives me a positive responce
on isset($WIT_CONF['URL']). In this case $WIT_CONF
is global and static. But I can also set up a $WIT_STATE
variable which will represent the state of the transaction.
Using the code of WIT_set_gv() and WIT_get_gv(), with the help
of a simple few lines (in my case: include globals.inc.php)
definition script I handle this problem.

In my case, for example, if 'WIT_STATE' (or other names)
is not a defined globally available variable I default to check
for a session variable.

For example you might warn or stop if a requested named variable
matches a $_POST, $_GET or $_SESSION variable name while you
do not expect so. i.e. all my private data has a wit_ prefix
but no public request has (shouldn't have) this prefix.

Oopss. I do realize that this comment might not be in the proper
place. i.e. "register_globals". Indeed it might give some advice
to users still using register_globals and willing to change the
code for a "better" solution. Of course
the simple switching to "register_globals = off" might not solve
the securities issues.

Cheers
F

Dice
16-Apr-2008 04:46

To expand on the nice bit of code Mike Willbanks wrote and Alexander tidied up, I turned the whole thing in a function that removes all the globals added by register_globals so it can be implemented in an included functions.php and doesn't litter the main pages too much.

```php
<?php
//Undo register_globals
function unregister_globals() {
  if (ini_get(register_globals)) {
    $array = array('_REQUEST', '_SESSION', '_SERVER', '_ENV', '_FILES');
    foreach ($array as $value) {
      foreach ($GLOBALS[$value] as $key => $var) {
        if ($var === $GLOBALS[$key]) {
          unset($GLOBALS[$key]);
        }
      }
    }
  }
}
?>
```

Ruquay K Calloway
01-Apr-2008 01:59

While we all appreciate the many helpful posts to get rid of register_globals, maybe you're one of those who just loves it. More likely, your boss says you just have to live with it because he thinks it's a great feature.

No problem, just call (below defined):

```php
<?php register_globals(); ?>
```

anywhere, as often as you want. Or update your scripts!

```php
<?php
/**
* function to emulate the register_globals setting in PHP
* for all of those diehard fans of possibly harmful PHP settings :-)
* @author Ruquay K Calloway
* @param string $order order in which to register the globals, e.g. 'egpcs' for default
*/
function register_globals($order = 'egpcs')
{
```

```
  // define a subroutine
  if(!function_exists('register_global_array'))
  {
    function register_global_array(array $superglobal)
    {
      foreach($superglobal as $varname => $value)
      {
        global $$varname;
        $$varname = $value;
      }
    }
  }

  $order = explode("\r\n", trim(chunk_split($order, 1)));
  foreach($order as $k)
  {
    switch(strtolower($k))
    {
      case 'e':    register_global_array($_ENV);        break;
      case 'g':    register_global_array($_GET);        break;
      case 'p':    register_global_array($_POST);        break;
      case 'c':    register_global_array($_COOKIE);    break;
      case 's':    register_global_array($_SERVER);    break;
    }
  }
}
?>
```

Tumasch
13-Dec-2007 01:50

In addition to Mike Willbanks post:

Put this to the beginning of every file or to a functions.inc.php and call it every time before start working with user variables.
This will prevent problems with wrong initalized variables or users who try to break your application.

And this has an extra bonus: Applications which still work are also register_globasl = off enabled!

```
<?php
//
// If register_globals is on, delete all variables exept the ones in the array
//
if (ini_get('register_globals')) {
  foreach ($GLOBALS as $int_temp_name => $int_temp_value) {
    if (!in_array($int_temp_name, array (
        'GLOBALS',
        '_FILES',
        '_REQUEST',
        '_COOKIE',
        '_SERVER',
        '_ENV',
        '_SESSION',
        ini_get('session.name'),
        'int_temp_name',
        'int_temp_value'
      ))) {
      unset ($GLOBALS[$int_temp_name]);
    }
  }
}
//
// Now, (re)import the variables
//
if (isset ($_REQUEST['pass']))
  $ext_pass = $_REQUEST['pass'];
if (isset ($_REQUEST['user']))
  $ext_user = $_REQUEST['user'];
if (isset ($_REQUEST['action']))
  $ext_action = $_REQUEST['action'];
//
// Cleanup entries
//
$int_pass = (isset ($ext_pass) ? preg_replace("'[^A-Z]'", "", $ext_pass) : '');
$int_user = (isset ($ext_user) ?
preg_replace("'[]A-Za-z0-9Ã¡Ã¤Ã Ã¢Ã£Ã«Ã¨Ã©Ãª Ã¯Ã¬Ã®Ã³Ã¶Ã²Ã´ÃµÃºÃ¼Ã¹Ã» \.^\$\!\_-()'", "", $ext_user)
: '');
$int_action = (isset ($ext_action) ? intval($ext_action) : '');
//
// Import Session variables
```

```
//
if (isset ($_SESSION)) {
  foreach ($_SESSION as $int_temp_key => $int_temp_value) {
    if ($int_temp_value != '') {
      $$int_temp_key = $int_temp_value;
    }
  }
}
//
// Import Cookie variables
//
if (isset ($_COOKIE)) {
  foreach ($_COOKIE as $int_temp_key => $int_temp_value) {
    if ($int_temp_value != '') {
      $$int_temp_key = $int_temp_value;
    }
  }
}
//
// From here on, work only with $int_ variables and you're safe!
//
?>

With this you can prevent a lot of different problems!
```

alan hogan
20-Jul-2007 03:08

```
Useful for shared hosting or scripts that you are sharing with other people.
<?php
// Effectively turn off dangerous register_globals without having to edit php.ini
if (ini_get(register_globals))  // If register_globals is enabled
{ // Unset $_GET keys
foreach ($_GET as $get_key => $get_value) {
  if (ereg('^([a-zA-Z]|_){1}([a-zA-Z0-9]|_)*$', $get_key)) eval("unset(\${$get_key});");
} // Unset $_POST keys
foreach ($_POST as $post_key => $post_value) {
  if (ereg('^([a-zA-Z]|_){1}([a-zA-Z0-9]|_)*$', $post_key)) eval("unset(\${$post_key});");

} // Unset $_REQUEST keys
foreach ($_REQUEST as $request_key => $request_value) {
  if (ereg('^([a-zA-Z]|_){1}([a-zA-Z0-9]|_)*$', $request_key)) eval("unset(\${$request_key});");
}
}
?>
```

dav at thedevelopersalliance dot com
18-Dec-2003 06:38

```
import_request_variables() has a good solution to part of this problem - add a prefix to all
imported variables, thus almost eliminating the factor of overriding internal variables through
requests. you should still check data, but adding a prefix to imports is a start.
```