PHP random string generator

Ask Question

I'm trying to create a randomized string in PHP, and I get absolutely no output with this:

```
<?php
function RandomString()
{
    $characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $randstring = '';
    for ($i = 0; $i < 10; $i++) {
        $randstring = $characters[rand(0, strlen($characters))];
    }
    return $randstring;
}
RandomString();
echo $randstring;
```

What am I doing wrong?

php string random

edited Apr 10 '16 at 8:35 Yogesh Suthar 26.2k 16 56 91

asked Dec 4 '10 at 22:56 Captain Lightning 3,420 4 13 14

186 My one line solution for generate short string is substr(md5(rand()), 0, 7); good luck ... – tasmaniski Jun 21 '12 at 14:46

4 @tasmaniski.. Your solution is ok.. But its less randomized! In your example the number of random strings that can be generated is limited by the size of integer. (2^32) at the max.. In case of the other solution, you can generate (62^8).. In case, I want larger strings, then number of distinct strings remain at max 2^32, but in the other solution it increases to (62^n).. – Manu Dec 23 '13 at 8:12

6 You forgot to add each new generated character to the string. You're just overwriting it as it is. Should be $randstring .= $characters.. – Spock Apr 25 '14 at 9:28

1 @CaptainLightning Can you please swap out the accepted answer for one of the more secure ones? :) – Scott Arciszewski Dec 8 '15 at 6:29

1 `strlen($characters)` => `strlen($characters) - 1` - string length starts with 1 – Zippp Apr 24 '17 at 13:07

47 Answers

1 2 next

To answer this question specifically, two problems:

1. `$randstring` is not in scope when you echo it.
2. The characters are not getting concatenated together in the loop.

Here's a code snippet with the corrections:

```
function generateRandomString($length = 10) {
    $characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $charactersLength = strlen($characters);
    $randomString = '';
    for ($i = 0; $i < $length; $i++) {
        $randomString .= $characters[rand(0, $charactersLength - 1)];
    }
    return $randomString;
}
```

Output the random string with the call below:

```
// Echo the random string.
// Optionally, you can give it a desired string length.
echo generateRandomString();
```

> **Please note that this generates predictable random strings. If you want to create secure tokens, see this answer.**

edited May 23 '17 at 12:34 Community ♦ 1 1

answered Dec 4 '10 at 22:57 Stephen Watkins 15.5k 8 48 90

3 Thanks for the code, but please change `rand()` to `mt_rand()` . I used your method and experienced a *ton* of collisions with names. – Nate Jul 25 '14 at 0:34

5 @FranciscoPresencia do you have any idea how horrifically inefficient that is? You are checking the length of a string twice per iteration! The strlen inside the loop should be cached in a variable before entering the loop as well. – developerbmw Nov 29 '14 at 6:51

4 This is not a good solution. The strings this generates will be predictable. :(– Scott Arciszewski Jun 29 '15 at 3:45

> Note: `str_shuffle()` internally uses `rand()` , which is unsuitable for cryptography purposes (e.g. generating random passwords). You want a secure random number generator instead. It also doesn't allow characters to repeat.

One more way.

UPDATED *(now this generates any length of string):*

```
function generateRandomString($length = 10) {
    return
substr(str_shuffle(str_repeat($x='0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTU
 ceil($length/strlen($x)) )),1,$length);
}

echo  generateRandomString();  // OR: generateRandomString(24)
```

That's it. :)

edited May 23 '17 at 12:10
Community ♦
1 1

answered Nov 3 '12 at 20:04
Pr07o7yp3
3,568 1 8 11

60 +1 for the shortest answer :). But not the best answer for every use-case. This code will output strings where each character won't appear more than once (thus making it weaker for brute force attacks) and won't output anything longer than 62 characters. – David Dec 5 '12 at 9:55

16 Do not do this, it is extremely weak. The longer you make your random string, the weaker it will be. – Abhi Beckert Feb 13 '13 at 21:38

19 Weak it may be, but it's quick and easy. Depending on the use-case, this is fine -- I've used it because I didn't need security or strength; just a quick-fire shuffle. I'm writing a unit test, and this gives me a suitably random input to test with. – SDC Mar 12 '13 at 13:44

20 @FranciscoPresencia the reason it is not secure is because it never uses the same character twice. That makes it a *terrible* password generator. Please everyone stop up voting this, it is totally insecure must never be used. As the password gets longer, the number of characters you must test in a brute force gets shorter because you do not bother testing any previously used character. – Abhi Beckert Aug 9 '13 at 19:33

7 @FranciscoPresencia: I support your comments about avoiding using this for random passwords. However, I disagree that this answer should not be plus-one'd. I have plus-one'd this option for since it is an effective one-line solution for generating random strings (the original topic of this question). – bwright Dec 6 '13 at 17:11

There are a lot of answers to this question, but none of them leverage a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG).

The simple, secure, and correct answer is to use RandomLib and don't reinvent the wheel.

For those of you who insist on inventing your own solution, PHP 7.0.0 will provide `random_int()` for this purpose; if you're still on PHP 5.x, we wrote a PHP 5 polyfill for `random_int()` so you can use the new API even before you upgrade to PHP 7.

Safely generating random integers in PHP isn't a trivial task. You should always check with your resident StackExchange cryptography experts before you deploy a home-grown algorithm in production.

With a secure integer generator in place, generating a random string with a CSPRNG is a walk in the park.

Creating a Secure, Random String

```
/**
 * Generate a random string, using a cryptographically secure
 * pseudorandom number generator (random_int)
 *                           to select from
 * @return string
 */
function random_str($length, $keyspace =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ')
{
    $pieces = [];
    $max = mb_strlen($keyspace, '8bit') - 1;
    for ($i = 0; $i < $length; ++$i) {
        $pieces []= $keyspace[random_int(0, $max)];
    }
    return implode('', $pieces);
}
```

Usage:

```
$a = random_str(32);
$b = random_str(8, 'abcdefghijklmnopqrstuvwxyz');
```

Demo: https://3v4l.org/b4PST (Ignore the PHP 5 failures; it needs random_compat)

edited Apr 1 at 20:29 answered Jun 29 '15 at 3:41

Scott Arciszewski
20.1k 8 49 128

3 Definitely, best answer if you are looking for random password generator! – Rafa0809 Dec 12 '16 at 13:13

2 should be accepted answer! – Ilario Engler Apr 10 '17 at 22:26

1 At the beginning of the function add `$keyspace = str_shuffle($keyspace );` for more security – Jevgenij Dmitrijev Apr 16 '17 at 19:17

6 What do you mean by "for more security"? We're already using a secure random number generator. – Scott Arciszewski Apr 17 '17 at 13:49

6 Using `random_int()` instead of `rand()` or `mt_rand()` adds no complexity for the developer. At the same time, it gives them greater security. People who come to StackOverflow looking for quick solutions might not know if the thing they're building needs to be secure or not. If we give them secure-by-default answers, they create a more secure Internet even if, from their perspective, it's totally accidental. Why you would oppose this goal is a mystery to me. – Scott Arciszewski Apr 1 at 20:21

Creates a 20 char long hexdec string:

```
$string = bin2hex(openssl_random_pseudo_bytes(10)); // 20 chars
```

In PHP 7 (random_bytes()):

```
$string = base64_encode(random_bytes(10)); // ~14 chars
// or
$string = bin2hex(random_bytes(10)); // 20 chars
```

edited Jun 30 '17 at 16:16 answered Sep 4 '14 at 18:25

Rudie
25.8k 29 109 156

7 Best one-liner ever! :) – tftd Sep 14 '15 at 0:53

Please note that `openssl_random_pseudo_bytes()` did not use a cryptographically strong algorithm until php 5.6. Related bug: bugs.php.net/bug.php?id=70014 – acidtv Aug 8 '16 at 15:30

Also note that the smallest string this can make is of length 2. If you pass a byte length of 1 or less to `openssl_random_pseudo_bytes()` you will get nothing back. Also, note that when using `bin2hex(openssl_random_pseudo_bytes($length / 2))` since you are working with integers, it will automatically remove the modulo and will use the next lowest multiple of 2. So, `$length = 19` will produce a string of length 18. – Fisc Oct 26 '16 at 21:56

Proposal to use it as file content that can be opened through `fgets($fp, 1024)` and every file editor that has problems with very long lines: `function string_rand($len, $split="\n") { return substr(chunk_split(bin2hex(openssl_random_pseudo_bytes(ceil($len / 2))), 1023, $split), 0, $len); }` Set `$split` to `null` if it should return a one-liner. By that you can use it for both cases. – mgutt Mar 27 '17 at 13:01

2 This should really be the accepted answer – Daniel Lo Nigro Jun 30 '17 at 4:43

@tasmaniski: your answer worked for me. I had the same problem, and I would suggest it for those who are ever looking for the same answer. Here it is from @tasmaniski:

```
<?php
    $random = substr(md5(mt_rand()), 0, 7);
    echo $random;
?>
```

edited Jul 26 '17 at 15:47 answered Feb 10 '13 at 8:24

Sagar V Humphrey

– The Surrican Sep 13 '14 at 14:23

4 Keep in mind `md5` will result in a 30 character limit and always lowercase characters. – fyrye Nov 3 '14 at 15:12

3 Additionally, rand() shouldn't be used for cryptography. If you're looking to generate a crypographically sound string, use openssl_random_pseudo_bytes. – mattkgross Feb 23 '15 at 4:44

`md5(rand())` only offers 2^32 possible values. This means after, on average, 2^16 random strings you will expect one to repeat. – Scott Arciszewski Jun 29 '15 at 3:45

1 Well..i would bet the OP is not talking about "securely random strings". This solution is compact, easy and pretty to remember, for the **right** use cases. And, if you need to take care of possible collisions, why not to prepend/append a timestamp to the random string? :) – Erenor Paz Dec 29 '16 at 14:10

Depending on your application (I wanted to generate passwords), you could use

```
$string = base64_encode(openssl_random_pseudo_bytes(30));
```

Being base64, they may contain `=` or `-` as well as the requested characters. You could generate a longer string, then filter and trim it to remove those.

`openssl_random_pseudo_bytes` seems to be the recommended way way to generate a proper random number in php. Why `rand` doesn't use `/dev/random` I don't know.

answered Feb 6 '13 at 17:40
rjmunro
17.4k 14 83 110

1 rand doesn't use /dev/urandom because that is only available in posix like environments and is not portable. – MacroMan Mar 25 '14 at 9:22

3 @MacroMan But `openssl_random_pseudo_bytes()` *is* portable. – Jack Jul 29 '14 at 5:29

If you want to strip out the extra base64 characters, try this: gist.github.com/zyphlar/7217f566fc83a9633959 – willbradley Dec 20 '14 at 22:38

1 This isn't wrong, but I would advise caution in how you discard the unwanted characters. See this pull request on the PHP league's OAuth2 Server, for example. – Scott Arciszewski Jul 6 '15 at 6:11

This is one of the best answers. Pity it doesn't have more support. I would recommend editing your answer for better handling of unwanted characters though. – jcuenod Aug 4 '16 at 18:32

I know this may be a bit late to the game, but here is a simple one-liner that generates a true random string without any script level looping or use of openssl libraries.

```
echo
substr(str_shuffle(str_repeat('0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVW
 mt_rand(1,10))),1,10);
```

To break it down so the parameters are clear

```
// Character List to Pick from
$chrList = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';

// Minimum/Maximum times to repeat character List to seed from
$chrRepeatMin = 1; // Minimum times to repeat the seed string
$chrRepeatMax = 10; // Maximum times to repeat the seed string

// Length of Random String returned
$chrRandomLength = 10;

// The ONE LINE random command with the above variables.
echo substr(str_shuffle(str_repeat($chrList,
mt_rand($chrRepeatMin,$chrRepeatMax))),1,$chrRandomLength);
```

This method works by randomly repeating the character list, then shuffles the combined string, and returns the number of characters specified.

You can further randomize this, by randomizing the length of the returned string, replacing `$chrRandomLength` with `mt_rand(8, 15)` (for a random string between 8 and 15 characters).

answered Apr 19 '14 at 21:18
Kraang Prime
4,452 4 27 64

sorry to say but when one character gots selected its probability of occuring again is not as high as the other characters still remaining in the pool. no true randomness here... – The Surrican Sep 13 '14 at 14:16

@TheSurrican - You would be incorrect in that statement. All characters have an equal probability using this algorithm, and thus is truly random. This is ensured by repeating the chrList string $chrRepeatMax, and the magic really happens in str_shuffle which determines the randomness. – Kraang Prime Apr 16 '15 at 8:56

All chars would appear only once and it is very prone to bruteforce guess – venimus Jul 20 '15 at 9:28