PHP – CSPRNG

The acronym CSPRNG stands for Cryptographically Secure Pseudorandom Number Generator. PHP function library includes many functions that generate random numbers. For example –

- **mt_rand()** – Generate a random value via the Mersenne Twister Random Number Generator
- **mt_srand()** – Seeds the Mersenne Twister Random Number Generator
- **rand()** – Generate a random integer.

Example

The following code shows how you can use the function **mt_rand()** to generate random numbers –

</> Open Compiler

```php
<?php
   # Generates random integer between the range
   echo "Random integer: " . rand(1,100) . PHP_EOL;
   # Generate a random value via the Mersenne Twister Random Number Generator
   echo "Random number: " . mt_rand(1,100);
?>
```

It will produce the following **output** –

```
Random integer: 45
Random number: 86
```

Note that the output may vary every time the code is executed. However, random numbers generated by these functions are not cryptographically safe, as it is possible to guess their outcome. PHP 7, introduced a couple of functions that generate secure random numbers.

The following functions which are cryptographically secure, are newly added –

- **random_bytes()** – Generates cryptographically secure pseudo-random bytes.
- **random_int()** – Generates cryptographically secure pseudo-random integers.

The random_bytes() Function

random_bytes() generates an arbitrary-length string of cryptographic random bytes that are suitable for cryptographic use, such as when generating salts, keys or initialization vectors.

```
string random_bytes ( int $length )
```

Parameters

- **length** – The length of the random string that should be returned in bytes.

The function returns a string containing the requested number of cryptographically secure random bytes.

If an appropriate source of randomness cannot be found, an Exception will be thrown. If invalid parameters are given, a TypeError will be thrown. If an invalid length of bytes is given, an Error will be thrown.

Example

Take a look at the following example –

</> Open Compiler

```
<?php
   $bytes = random_bytes(5);
   print(bin2hex($bytes));
?>
```

It may produce the following **output** (it may differ every time) –

```
6a85eec950
```

The random_int() Function

random_int() generates cryptographic random integers that are suitable for use where unbiased results are critical.

```
int random_int ( int $min , int $max )
```

Parameters

- **min** – The lowest value to be returned, which must be PHP_INT_MIN or higher.
- **max** – The highest value to be returned, which must be less than or equal to PHP_INT_MAX.

The function returns a cryptographically secure random integer in the range min to max, inclusive.

If an appropriate source of randomness cannot be found, an Exception will be thrown. If invalid parameters are given, a TypeError will be thrown. If max is less than min, an Error will be thrown.

Example

Take a look at the following example –

</> Open Compiler

```
<?php
   print(random_int(100, 999));
   print("\n");
   print(random_int(-1000, 0));
?>
```

It may produce the following **output** (it differs every time) –

```
495
-563
```