

Forum: [Hack Forums](#)

Thread: [Need help cracking seed](#)

explosivepastry - August 8, 2021, 8:45 am

Hello!

So I'm doing a code review for someone and saw this code in the sourcecode:

and its being used for the password reset token. Is there any way to crack this as I know str_shuffle is not supposed to be used as crypto.

If so, can someone point me in the right direction?

Thanks!

guardian - August 22, 2021, 5:04 am

I don't see any type of encryption/hashing here, thus there is nothing to crack.

Edit: I read it a bit too fast and got so much into writing this post that I kind of forgot to check it out again lol. I guess you are referring to the scenario of an attacker cracking the token? My above statement was written really fast, of course you can crack something that is not encrypted or hashed, i.e. an account, a program etc, just worked too much with crypto stuff lately lol.

If you mean, if there would be a way for an attacker to find out the generated password reset token (crack it), I would say yes and no. By that I mean that technically it would be possible, although with today's technical possibilities it would be close to impossible, because the generated password reset token is made up of 60 randomly concatenated characters of the characters specified inside str_shuffle, it will be possible, if not even easy, to crack it with future tech. But it will take some time until this is the case.

To explain the code a little bit:

\$key: variable that the password reset key will be stored in

substr: a function with 3 parameters: string, offset, length; the function 'cuts out' a (sub)string from the string given as an argument, starting at offset's position, in our case 0 (in programming we always start counting from 0), then going on until either length (in our case 60) is met or the string, given as an argument, ends. Length is the maximum length, but if the string is shorter than length, it ends before length is met.

str_shuffle: a function that has one paramter: string; you simply give it a string and it will shuffle all the characters inside it and return the new string. Note that it will really shuffle all the characters, so if you put in 10 chars, your output will always contain the same 10 chars. Each character will only be used a single time inside the shuffle function. All it does is change the order. In our example the str_shuffle function's return value is used as the string argument within substr.

The whole process works like this:

`str_shuffle` shuffles the string `'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'` and returns the shuffled string, which is then taken as an argument by `substr`. `substr` goes on and takes the first 60 characters of the string returned by `str_shuffle`. Afterwards the variable `$key` holds the value returned by `substr`: the first 60 characters of the string returned by `str_shuffle`.

Let's have a little test run:

The output is:

Code `R3aJxebmFQjNpY8fIBh2glKi5MP19kvcAErS7GnHOsV6wCTdLz4ZD0WotqUu`

A 60 character string that contains the first 60 characters of the shuffled string returned by `str_shuffle`.

Concerning randomness of `str_shuffle`, this [Stack Overflow post](#) might interest you.

As explained above, it only uses the characters specified by you and each character is only used once.

Additionally, I would recommend you to have a look at [mt_rand](#).