Sec Bug #81744 Password_verify() always return true with some hash

Submitted: 2023-01-05 12:52 UTC **Modified:** 2023-02-13 04:40 UTC
From: tech at mkdgs dot fr **Assigned:** stas (profile)
Status: Closed **Package:** *Encryption and hash functions
PHP Version: 8.2.0 **OS:**
Private report: No **CVE-ID:** 2023-0567

View | Developer | Edit

[2023-01-05 12:52 UTC] tech at mkdgs dot fr

```
Description:
------------
Password_verify() always return true with some hash.
I able to login without password when password_verify is used and a bad hash is stored in database.

see test script for a evil hash example.
There is other forms of magic hash with the same consequence.

seen on many version of php on linux (maybe present on other system).
perhaps a bigger security problem in the underlying code.


Test script:
---------------
<?php
$pass_rand = rand(1,999).'test'.rand(1,999);
echo $pass_rand. ':';
echo (password_verify($pass_rand , '$2y$10$am$2y$10$am')) ? 'OK' : 'FAIL';;
```

Patches

Pull Requests

Pull requests:

- Ignore externally managed and generated files (web-windows/21)

History

All | Comments | Changes | Git/SVN commits | Related reports

[2023-01-05 13:05 UTC] ondrej@php.net

```
The documentation for `password_verify()` clearly states that the `$hash` has to be valid:

> hash
> A hash created by password_hash().

I fail to see how this is a bug if you feed it garbage; also the documentation to crypt states:

> Using characters outside of this range in the salt will cause crypt() to return a zero-length string. The
two digit cost parameter is the base-2 logarithm of the iteration count for the underlying Blowfish-based
hashing algorithm and must be in range 04-31, values outside this range will cause crypt() to fail.
```

PHP clearly behaves as described.

[2023-01-06 10:58 UTC] tech at mkdgs dot fr

Sorry, i don't clearly see it in documentation:
https://www.php.net/manual/en/function.password-verify.php

I don't think this is clear (and maybe for many developer too)
Many solution fail with that if an attacker put this kind of hash in db
(via sql injection)

At least clearly update the documentation about this point.
And i think, but it's my point of view, this function must be simple as possible
too avoid security problem.

Ok, it's not a bug it's a feature, but for now you can validate an access with a bad hash, it's not very intuitive.
I agree, the hash is not valid, but why not checking the hash ??
If it's for performance reason, let an option for not checking it.
But make this function simple.

```
<?php
var_dump(PHP_OS);
$pass_rand = rand(1,999).'test'.rand(1,999);
echo (password_verify($pass_rand , '$2y$10$am$2y$10$am')) ? 'Password match hash' : 'FAIL Password not match hash';
echo "\r";
echo ( password_hash($pass_rand, PASSWORD_DEFAULT) === '$2y$10$am$2y$10$am' ) ? 'OK: Hash match' : 'Fail: Hash not match';
```

[2023-01-23 14:20 UTC] cmb@php.net

From the RFC which introduced the password functions[1]:

```
| When passed a correct password and the generated hash from
| password_hash(), the function will return a boolean true. If there
| is any failure (hash is invalid, password is incorrect, hash is
| corrupted, etc), the function will return a boolean false.
```

So according to that statement, the current behavior is a bug.

> […], values outside this range will cause crypt() to fail.

Hmm:

```
php > var_dump(crypt("foo", '$2y$10$am$2y$10$am'));
string(18) "$2y$10$am$2y$10$am"
```

How is the programmer supposed to know that this is a failure
return? Ah, the return values section clarifies:

```
| Returns the hashed string or a string that is shorter than 13
| characters and is guaranteed to differ from the salt on failure.
```

Okay. Still, I fail to see why password_verify() returns true,
while crypt() fails for the same broken salt/hash. Furthermore,
it seems that password_verify() fails for broken argon2 hashes.

And when I look at the implementation of php_password_bcrypt_verify()
in ext/standard/password.c:

```
        zend_string *ret = php_crypt(ZSTR_VAL(password), (int)ZSTR_LEN(password), ZSTR_VAL(hash),
(int)ZSTR_LEN(hash), 1);

        if (!ret) {
                return 0;
        }

        if (ZSTR_LEN(hash) < 13) {
                zend_string_free(ret);
                return 0;
        }
```

I see that we are checking whether the length is less than 13
characters, but we do not check that the hash is returned
unmodified.

[1] <https://wiki.php.net/rfc/password_hash>

[2023-01-23 21:00 UTC] cmb@php.net

There is now respective advisory at
<https://github.com/php/php-src/security/advisories/GHSA-7fj2-8x79-rjf4>.

tech at mkdgs dot fr, if you have a GH account, I can add you as collaborator for that ticket.

[2023-01-27 07:54 UTC] stas@php.net

```
-CVE-ID:
+CVE-ID: needed
```

[2023-01-29 07:46 UTC] stas@php.net

```
-CVE-ID: needed
+CVE-ID: 2023-0567
```

[2023-02-13 04:40 UTC] stas@php.net

```
-Status: Open
+Status: Closed
-Assigned To:
+Assigned To: stas
```

[2023-02-13 04:40 UTC] stas@php.net

The fix for this bug has been committed.
If you are still experiencing this bug, try to check out latest source from https://github.com/php/php-src
and re-test.
Thank you for the report, and for helping us make PHP better.

[2023-03-13 11:21 UTC] cogaininhhoa87 at gmail dot com

```
$password = '$2a$07$58a5710f2225d$$$$$$$$$.2AucaR7lucHAhCAnW0VUPNgiajDdVkO';
$pattern = '/(^\$.+\$).+/';
preg_match($pattern, $password, $matches);
if (empty($matches[1])) {
    return false;// Unknown salt pattern.
}
$hash = crypt('test123', $matches[1]);
var_dump($hash);die;
```

```
=> it's return *0

Dear support, I had a script code above. With 8.0.28, it returned an error instead of an encrypted string.
Could you help to fix that? Thanks!
```

[2023-03-13 12:18 UTC] tech at mkdgs dot fr

```
cmb@php.net, here is my GH email: md-github@mkdgs.fr

Sorry for the delay, I didn't get a notification about the change here.
I only received the last one, right now, with the last comment.
```

[2023-04-27 05:41 UTC] shipramondal242008 at gmail dot com

```
The following pull request has been associated:

Patch Name: Ignore externally managed and generated files
On GitHub:  https://github.com/php/web-windows/pull/21
Patch:      https://github.com/php/web-windows/pull/21.patch
```