Happy Hacking!
Exploit Dev & Web App Security

CVE-2020-8088 – UseBB Forum 1.0.12 – PHP Type Juggling vulnerability

Posted on January 22, 2020 by Xavi

Hello!

Last week I was reading about PHP Type Juggling vulnerabilities and I decided to spend a couple of days learning about them.

These vulnerabilities can happen during comparison of PHP variables, because PHP will automatically convert the data into a common comparable type.

My idea was to try to find one by my own. But first I needed to look for some PHP open source code to review.

I thought that I could find one in old open source forums. My idea was to try to understand the authentication and the password recovery implementations.

After installing a couple of different open source forums I've found UseBB software that seemed to have an interesting implementation of the login.

Installing the software and creating and admin user

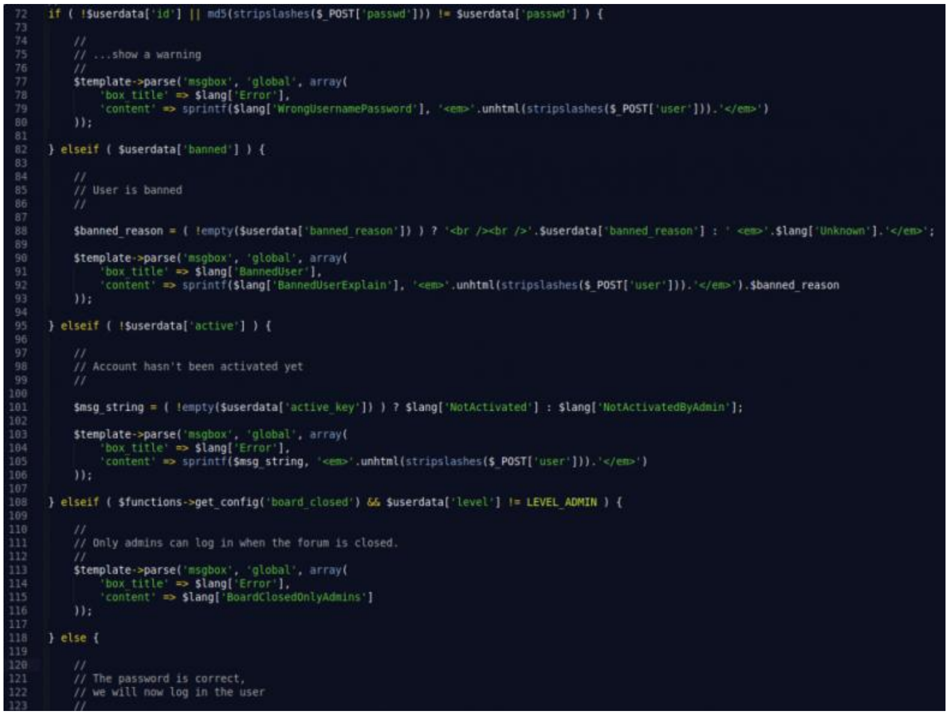
So I installed the software, to do that I created a database and followed the installation steps.

I created an admin user with the following credentials:

```
username=admin
password=aabC9RqS
```

Checking the login implementation

Doing a quick code check, I've found that the login was implemented in the file: "/sources/panel_login.php"



```
if ( !$userdata['id'] || md5(stripslashes($_POST['passwd'])) != $userdata['passwd'] ) {

    //
    // ...show a warning
    //
    $template->parse('msgbox', 'global', array(
        'box_title' => $lang['Error'],
        'content' => sprintf($lang['WrongUsernamePassword'], '<em>'.unhtml(stripslashes($_POST['user'])).'</em>')
    ));

} elseif ( $userdata['banned'] ) {

    //
    // User is banned
    //

    $banned_reason = ( !empty($userdata['banned_reason']) ) ? '<br /><br />'.$userdata['banned_reason'] : ' <em>'.$lang['Unknown'].'</em>';

    $template->parse('msgbox', 'global', array(
        'box_title' => $lang['BannedUser'],
        'content' => sprintf($lang['BannedUserExplain'], '<em>'.unhtml(stripslashes($_POST['user'])).'</em>').$banned_reason
    ));

} elseif ( !$userdata['active'] ) {

    //
    // Account hasn't been activated yet
    //

    $msg_string = ( !empty($userdata['active_key']) ) ? $lang['NotActivated'] : $lang['NotActivatedByAdmin'];

    $template->parse('msgbox', 'global', array(
        'box_title' => $lang['Error'],
        'content' => sprintf($msg_string, '<em>'.unhtml(stripslashes($_POST['user'])).'</em>')
    ));

} elseif ( $functions->get_config('board_closed') && $userdata['level'] != LEVEL_ADMIN ) {

    //
    // Only admins can log in when the forum is closed.
    //
    $template->parse('msgbox', 'global', array(
        'box_title' => $lang['Error'],
        'content' => $lang['BoardClosedOnlyAdmins']
    ));

} else {

    //
    // The password is correct,
    // we will now log in the user
    //
```

UseBB Forum Login implementation

Identifying a vulnerability

The application does different checks to verify if the password supplied by the user is correct. The most important line for checking the Type Juggling vulnerability is the following:

if (!$userdata['id'] || **md5**(stripslashes($_POST['passwd'])) **!=** $userdata['passwd']) {

Notice that it’s using only one equal sign, that is a loose comparison, and they should have used an strict one.

In this link you can read the following:

https://www.whitehatsec.com/blog/magic-hashes/

“*For more than the last decade, PHP programmers have been wrestling with the equals-equals (==) operator. It’s caused a lot of issues. This has a particular implication for password hashes. Password hashes in PHP are base16 encoded and can come in the form of “0e812389…”. The problem is in == comparison the 0e means that if the following characters are all digits the whole string gets treated as a float.* “

What they are talking about, is that when there is a loose comparison, you can do strange things, like this:

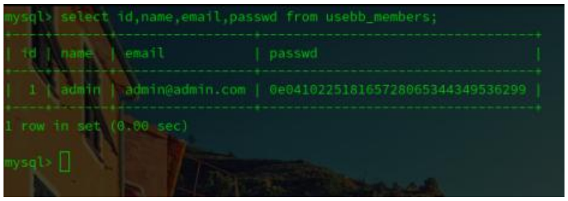
```
socket@lab:~$ php -r "print md5('aabC9RqS');";echo ''
0e041022518165728065344349536299
socket@lab:~$ php -r "print md5('aabg7XSs');";echo ''
0e087386482136013740957780965295
socket@lab:~$ php -r "var_dump(md5('aabC9RqS') == md5('aabg7XSs'));"
bool(true)
```

As you can see the hashes are different but when we compare them with a loose comparison the result is true.

Login with the same user using a different password

Before doing anything, let’s check the current status of our database. Specifically the table usebb_members that stores usernames and hashed passwords.

I see the following hash stored as the password:



```
mysql> select id,name,email,passwd from usebb_members;
+----+-------+-----------------+----------------------------------+
| id | name  | email           | passwd                           |
+----+-------+-----------------+----------------------------------+
|  1 | admin | admin@admin.com | 0e041022518165728065344349536299 |
+----+-------+-----------------+----------------------------------+
1 row in set (0.00 sec)

mysql> 
```

UseBB Forum admin password hash

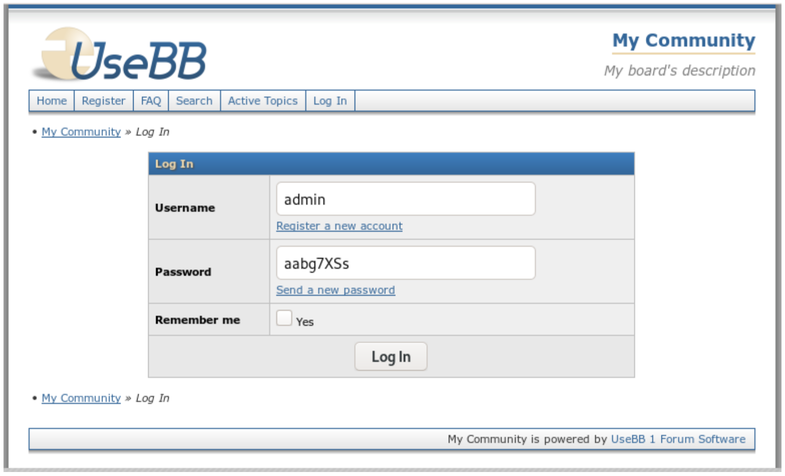
If we remember the login verification, this hash is the value for the variable: $userdata[‘passwd’]

Doing a quick verification we can see that this hash, is the md5 value of the password that we used when we registered the user:

```
socket@lab:~$ php -r "print md5('aabC9RqS');";echo ''
0e041022518165728065344349536299
```

We know that the password for the user admin is: “aabC9RqS” but let’s try to use “aabg7XSs” instead.

We try to login using this password:



Log In

Username	admin Register a new account
Password	aabg7XSs Send a new password
Remember me	□ Yes

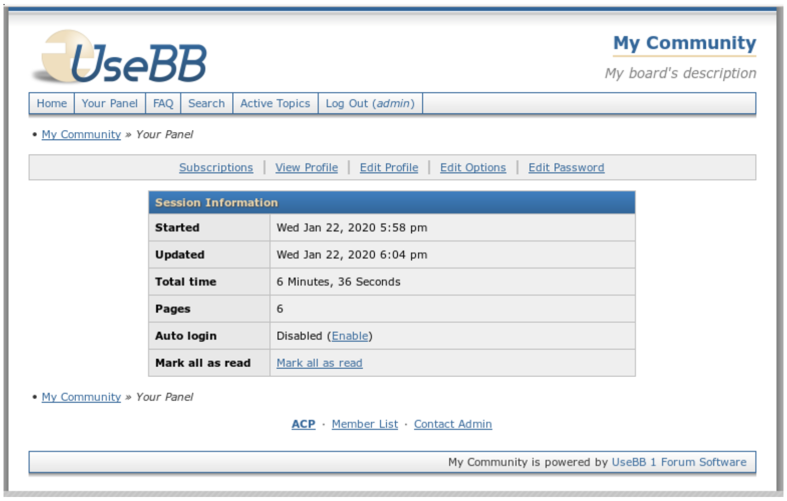
The server is evaluating this:

```
md5('aabC9RqS') == md5('aabg7XSs')
0e041022518165728065344349536299 == 0e087386482136013740957780965295
```

And as we saw before…

```
php -r "var_dump(md5('aabC9RqS') == md5('aabg7XSs'));"
bool(true)
```

So, we are in 🙂



Session Information	
Started	Wed Jan 22, 2020 5:58 pm
Updated	Wed Jan 22, 2020 6:04 pm
Total time	6 Minutes, 36 Seconds
Pages	6
Auto login	Disabled (Enable)
Mark all as read	Mark all as read

My Community is powered by UseBB 1 Forum Software

Vulnerability solution:

We need to add an extra equal in the line 72 of sources/panel_login.php

if (!$userdata[‘id’] || md5(stripslashes($_POST[‘passwd’])) **!==** $userdata[‘passwd’]) {

This software seems to doesn’t have support. But if you are using it, I recommend you to migrate it to Drupal using this plugin:

https://www.drupal.org/project/usebb2drupal

Interesting resources:

If you are interested reading more about this topic I recommend you some resources:

https://www.whitehatsec.com/blog/magic-hashes/

https://www.owasp.org/images/6/6b/PHPMagicTricks-TypeJuggling.pdf

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Type%20Juggling

Thank you for reading the blog! See you soon 🙂

This entry was posted in Hacking Web and tagged CVE, forum, Hacking web, php, type juggling, usebb, vulnerability discovery. Bookmark the permalink.

Happy Hacking!

Proudly powered by WordPress.