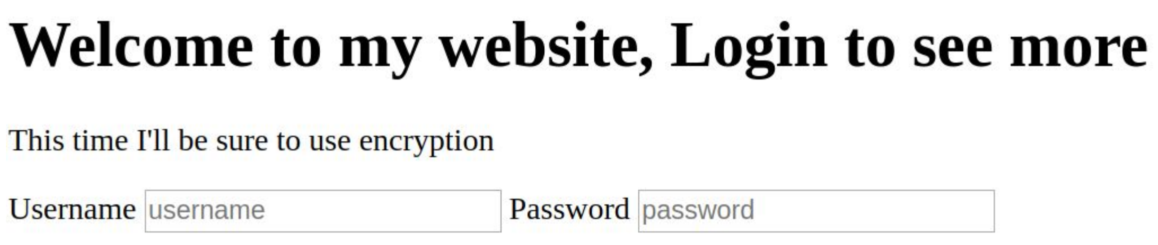
WEB: Ajax Not Borax

The webpage looks like the following:



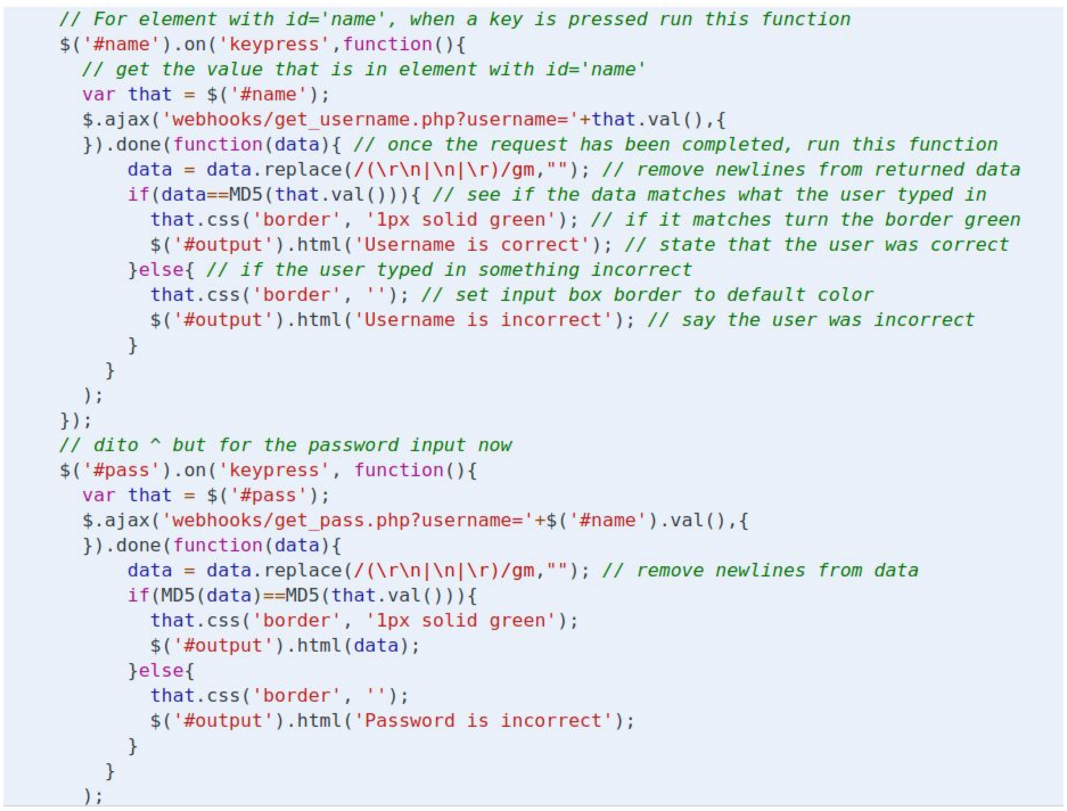
Welcome to my website, Login to see more

This time I'll be sure to use encryption

Username username Password password

Our goal is probably to find a proper *username* and *password* that gives us access to the system and prints us (?) a flag. If you play a bit by inserting random strings inside the *textboxes* you see a message that says “*username incorrect*” or “*password incorrect*”.

We can analyze the *javascript.*



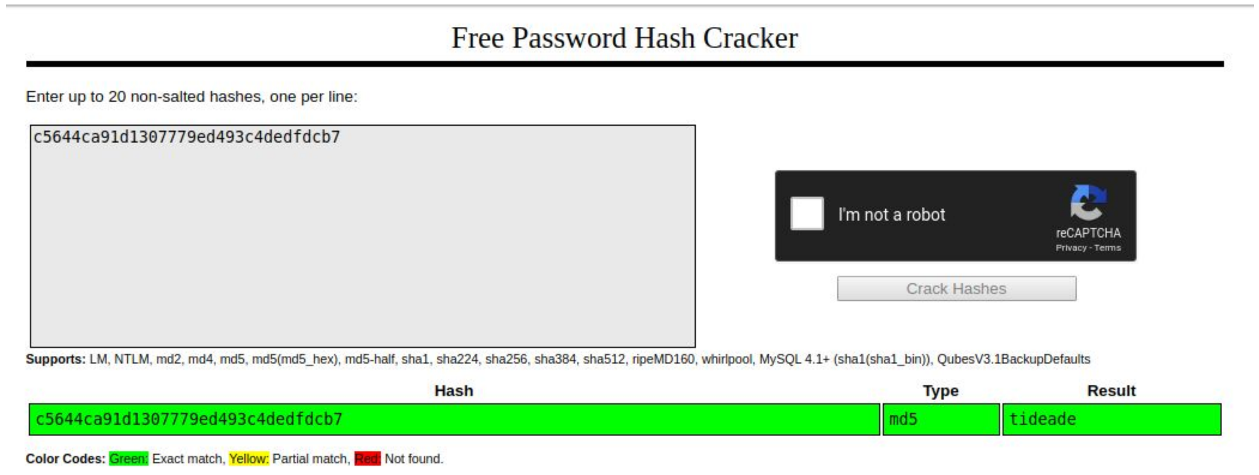
```
// For element with id='name', when a key is pressed run this function
$('#name').on('keypress',function(){
  // get the value that is in element with id='name'
  var that = $('#name');
  $.ajax('webhooks/get_username.php?username='+that.val(),{
  }).done(function(data){ // once the request has been completed, run this function
      data = data.replace(/(\r\n|\n|\r)/gm,""); // remove newlines from returned data
      if(data==MD5(that.val())){ // see if the data matches what the user typed in
        that.css('border', '1px solid green'); // if it matches turn the border green
        $('#output').html('Username is correct'); // state that the user was correct
      }else{ // if the user typed in something incorrect
        that.css('border', ''); // set input box border to default color
        $('#output').html('Username is incorrect'); // say the user was incorrect
      }
    }
  );
});
// dito ^ but for the password input now
$('#pass').on('keypress', function(){
  var that = $('#pass');
  $.ajax('webhooks/get_pass.php?username='+$('#name').val(),{
  }).done(function(data){
      data = data.replace(/(\r\n|\n|\r)/gm,""); // remove newlines from data
      if(MD5(data)==MD5(that.val())){
        that.css('border', '1px solid green');
        $('#output').html(data);
      }else{
        that.css('border', '');
        $('#output').html('Password is incorrect');
      }
    }
  );
```

There are two main functions, one that checks the *username*, and one that checks the *password*. The correct pairs of username-passwords are retrieved using a “webhooks” *ajax* function. These values are then compared with the MD5 function.

In the browser, we can set a breakpoint in the line that assigns “data” in the first *if-block*; the MD5 of the username (real) is:

"c5644ca91d1307779ed493c4dedfdcb7"

Can we decrypt it? Well, there are tons of crackers online, for example we can use the following link.


Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
c5644ca91d1307779ed493c4dedfdcb7
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
c5644ca91d1307779ed493c4dedfdcb7	md5	tideade

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

The username is “tideade”, and if we insert it in our interface we see that it’s the right one.

Let’s focus on the second *if-block*. Now the comparison is between 2 MD5 values, i.e., the retrieved real password is in clear! With the debugger we obtain the following value:

"ZmxhZ3tzZDkwSjBkbkxLSjFsczlISmVkfQ=="

It’s a bese64, but we don’t care. If we insert it in the field, a message appears: the password!
We can decrypt it and convert it into a normal base.
The decoding reveals the flag:

flag{sd90J0dnLKJ1ls9HJed}