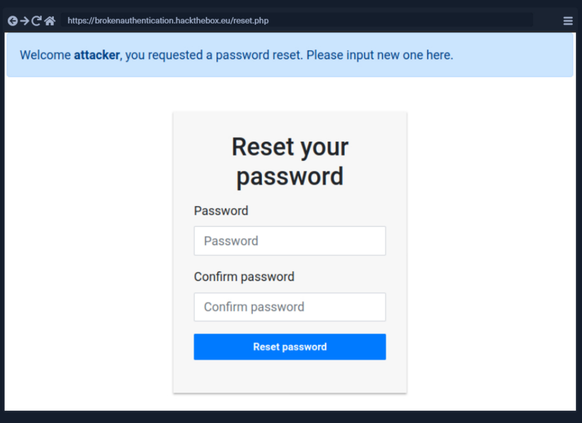
Username Injection

When trying to understand the high-level logic behind a reset form, it is unimportant if it sends a token, a temporary password, or requires the correct answer. At a high level, when a user inputs the expected value, the reset functionality lets the user change the password or pass the authentication phase. The function that checks if a reset token is valid and is also the right one for a given account is usually carefully developed and tested with security in mind. However, it is sometimes vulnerable during the second phase of the process, when the user resets the password after the first login has been granted.

Imagine the following scenario. After creating an account of our own, we request a password reset. Suppose we come across a form that behaves as follows.



We can try to inject a different username and/or email address, looking for a possible hidden input value or guessing any valid input name. It has been observed that some applications give precedence to received information against information stored in a session value.

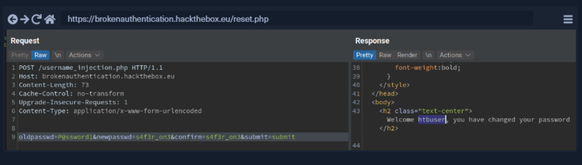
An example of vulnerable code looks like this (the `$_REQUEST` variable contains both `$_GET` and `$_POST`):

Code: php

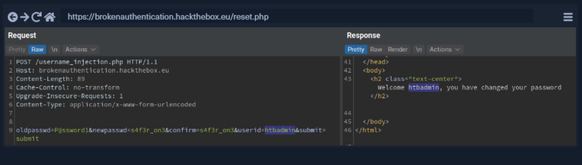
```php
<?php
  if isset($_REQUEST['userid']) {
    $userid = $_REQUEST['userid'];
  } else if isset($_SESSION['userid']) {
    $userid = $_SESSION['userid'];
  } else {
    die("unknown userid");
  }
```

This could look weird at first but think about a web application that allows admins or helpdesk employees to reset other users' passwords. Often, the function that changes the password is reused and shares the same codebase with the one used by standard users to change their password. An application should always check authorization before any change. In this case, it has to check if the user has the grants to modify the password for the target user. Often, the function that changes the password is reused and shares the same codebase with the one used by standard users to change their password. With this in mind, you should enumerate the web application to identify how it expects the username or email field during the login phase, when there are messages or communication exchange, or when you see other users' profiles. Having collected a list of all possible input field names, we will attack the application. The attack will be executed by sending a password reset request while logged in with our user and injecting the target user's email or username through the possible field names (one at a time).

We brute-forced the username and password on a web application that uses `userid` as a field name during the login process in previous exercises. Let us keep this field as an identifier of the user and operate on it. A standard request looks as follows.



If you tamper with the request by adding the `userid` field, you can change the password for another user.



As we can see, the application replies with a `success` message.

When we have a small number of fields and user/email values to test, you can mount this attack using an intercepting proxy. If you have many of them, you can automate the attack using any fuzzer or a custom script. We prepared a small playground to let you test this attack. You can download the PHP script here and Python script here. Take your time to study both files, then try to replicate the attack we showed.

Start Instance

∞ / 1 spawns left

Waiting to start...

Questions

Answer the question(s) below to complete this Section and earn cubes!

Cheat Sheet

Target: Click here to spawn the target system!

+2 Login with the credentials "htbuser:htbuser" and abuse the reset password function to escalate to "htbadmin" user. What is the flag?

HTB{us3rn4m3_1nj3ct3d}

Submit Hint

Previous Next

Mark Complete & Next

Cheat Sheet

Resources

Go to Questions

Table of Contents

Broken Authentication

What is Authentication

Overview of Authentication Methods

Overview of Attacks Against Authentication

Login Bruteforcing

Default Credentials

Weak Bruteforce Protections

Bruteforcing Usernames

Bruteforcing Passwords

Predictable Reset Token

Password Attacks

Authentication Credentials Handling

Guessable Answers

Username Injection

Session Attacks

Bruteforcing Cookies

Insecure Token Handling

Skill Assessment

Skill Assessment - Broken Authentication

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left