

PHPAuth

PHPAuth is a secure PHP Authentication class that easily integrates into any site.

[View on GitHub](#) [Download .zip](#) [Download .tar.gz](#)

build passing

PHPAuth

What is it

PHPAuth is a secure user authentication class for PHP websites, using a powerful password hashing system and attack blocking to keep your website and users secure.

Features

- Authentication by email and password combination
- Uses [bcrypt](#) to hash passwords, a secure algorithm that uses an expensive key setup phase
- Uses an individual 128 bit salt for each user, pulled from /dev/urandom, making rainbow tables useless
- Uses PHP's [PDO](#) database interface and uses prepared statements meaning an efficient system, resilient against SQL injection
- Blocks (or verifies) attackers by IP for any defined time after any amount of failed actions on the portal
- No plain text passwords are sent or stored by the system
- Integrates easily into most existing websites, and can be a great starting point for new projects
- Easy configuration of multiple system parameters
- Allows sending emails via SMTP or sendmail
- Blocks disposable email addresses from registration

User actions

- Login
- Register
- Activate account
- Resend activation email
- Reset password
- Change password
- Change email address
- Delete account
- Logout

Requirements

- PHP 5.4
- MySQL / MariaDB database or PostgreSQL database

Composer Support

PHPAuth can now be installed with the following command:

```
composer require phpauth/phpauth
```

Then: `require 'vendor/autoload.php';`

Configuration

The database table `config` contains multiple parameters allowing you to configure certain functions of the class.

- `site_name` : the name of the website to display in the activation and password reset emails
- `site_url` : the URL of the Auth root, where you installed the system, without the trailing slash, used for emails.
- `site_email` : the email address from which to send activation and password reset emails
- `site_key` : a random string that you should modify used to validate cookies to ensure they are not tampered with
- `site_timezone` : the timezone for correct datetime values
- `site_activation_page` : the activation page name appended to the `site_url` in the activation email
- `site_password_reset_page` : the password reset page name appended to the `site_url` in the password reset email
- `cookie_name` : the name of the cookie that contains session information, do not change unless necessary
- `cookie_path` : the path of the session cookie, do not change unless necessary
- `cookie_domain` : the domain of the session cookie, do not change unless necessary
- `cookie_secure` : the HTTPS only setting of the session cookie, do not change unless necessary
- `cookie_http` : the HTTP only protocol setting of the session cookie, do not change unless necessary
- `cookie_remember` : the time that a user will remain logged in for when ticking "remember me" on login. Must respect PHP's [strtotime](#) format.
- `cookie_forget` : the time a user will remain logged in when not ticking "remember me" on login. Must respect PHP's [strtotime](#) format.
- `bcrypt_cost` : the algorithmic cost of the bcrypt hashing function, can be changed based on hardware capabilities
- `smtp` : 0 to use sendmail for emails, 1 to use SMTP
- `smtp_host` : hostname of the SMTP server
- `smtp_auth` : 0 if the SMTP server doesn't require authentication, 1 if authentication is required
- `smtp_username` : the username for the SMTP server
- `smtp_password` : the password for the SMTP server
- `smtp_port` : the port for the SMTP server
- `smtp_security` : NULL for no encryption, `tls` for TLS encryption, `ssl` for SSL encryption
- `verify_password_min_length` : minimum password length, default is 3
- `verify_password_max_length` : maximum password length, default is 150
- `verify_password_strong_requirements` : use strong password requirements (at least one uppercase and lowercase character, and at least one digit), default is 1 (true)
- `verify_email_min_length` : minimum EMail length, default is 5
- `verify_email_max_length` : maximum EMail length, default is 100
- `verify_email_use_banlist` : use banlist while checking allowed EMail (see `/files/domains.json`), default is 1 (true)
- `attack_mitigation_time` : time used for rolling attempts timeout, default is +30 minutes. Must respect PHP's [strtotime](#) format.
- `attempts_before_verify` : maximum amount of attempts to be made within `attack_mitigation_time` before requiring captcha. Default is 5
- `attempt_before_block` : maximum amount of attempts to be made within `attack_mitigation_time` before temporally blocking the IP address. Default is 30
- `password_min_score` : the minimum score given by [zxcvbn](#) that is allowed. Default is 3

The rest of the parameters generally do not need changing.

CAPTCHA Implementation

If `isBlocked()` returns `verify`, then a CAPTCHA code should be displayed. The method `checkCaptcha($captcha)` is called to verify a CAPTCHA code. By default this method returns `true`, but should be overridden to verify a CAPTCHA.

For example, if you are using Google's ReCaptcha NoCaptcha, use the following code:

```
private function checkCaptcha($captcha)
{
    try {

        $url = 'https://www.google.com/recaptcha/api/siteverify';
        $data = ['secret' => 'your_secret_here',
            'response' => $captcha,
            'remoteip' => $_SERVER['REMOTE_ADDR']];

        $options = [
            'http' => [
                'header' => "Content-type: application/x-www-form-urlencoded\r\n",
                'method' => 'POST',
                'content' => http_build_query($data)
            ]
        ];

        $context = stream_context_create($options);
        $result = file_get_contents($url, false, $context);
        return json_decode($result)->success;
    }
    catch (\Exception $e) {
        return false;
    }
}
```

If a CAPTCHA is not to be used, please ensure to set `attempt_before_block` to the same value as `attempts_before_verify`.

How to secure a page

Making a page accessible only to authenticated users is quick and easy, requiring only a few lines of code at the top of the page:

```
<?php

include("Config.php");
include("Auth.php");

$dbh = new PDO("mysql:host=localhost;dbname=phpauth", "username", "password");

$config = new PHPAuth\Config($dbh);
$auth = new PHPAuth\Auth($dbh, $config);

if (!$auth->isLogged()) {
    header('HTTP/1.0 403 Forbidden');
    echo "Forbidden";

    exit();
}

?>
```

Message languages

The language for error and success messages returned by PHPAuth can be configured by passing in one of the available languages as the third parameter to the Auth constructor. If no language parameter is provided then the default en_GBlanguage is used.

Example: `$auth = new PHPAuth\Auth($dbh, $config, "fr_FR");`

Available languages:

- en_GB (Default)
- de_DE
- fa_IR
- fr_FR
- it_IT
- nl_BE
- pt_BR
- ru_RU

Documentation

All class methods are documented in [the Wiki](#)
System error codes are listed and explained [here](#)

Contributing

Anyone can contribute to improve or fix PHPAuth, to do so you can either report an issue (a bug, an idea...) or fork the repository, perform modifications to your fork then request a merge.

Credits

- [password_compat](#) - [@ircmaxell](#)
- [disposable](#) - [@lavab](#)
- [PHPMailer](#) - [@PHPMailer](#)
- [zxcvbn-php](#) - [@bjeavons](#)

[PHPAuth](#) is maintained by [PHPAuth](#). This page was generated by [GitHub Pages](#) using the [Cayman theme](#) by [Jason Long](#).