

Security



- [Initial Security Setup](#)
- [Multi-Factor Authentication](#)
- [Securing Images](#)
 - [Alternative Storage Options](#)
 - [Complex Urls](#)
 - [Prevent Directory Indexes](#)
- [Attachments](#)
- [Filesystem Permissions](#)
- [User Passwords](#)
- [JavaScript in Page Content](#)
- [Web Crawler Control](#)
- [Secure Cookies](#)
- [Host Iframe Control](#)
- [Iframe Source Control](#)
- [Failed Access Logging](#)
- [Untrusted Server Side Requests](#)
- [Server Side Request Allow List](#)
- [Content Security Policy \(CSP\)](#)
- [MySQL SSL Connection](#)
- [Using BookStack Content Externally](#)

Since BookStack can hold important information for users you should be aware of any potential security concerns. Read through the below to ensure you have secured your BookStack instance. Note, The below only relates to BookStack itself. The security of the server BookStack is hosted on is not instructed below but should be taken into account.

If you'd like to be notified of new potential security concerns you can sign-up to the [BookStack security mailing list](#). For reporting security vulnerabilities, please see the ["Security" section](#) of the project readme on GitHub.

Initial Security Setup

1. Ensure you change the password and email address for the initial `admin@admin.com` user.
 2. Ensure only the `public` folder is being served by your webserver. Serving files above this folder opens up a lot of code that does not need to be public. Triple check this if you have installed BookStack within the commonly used `/var/www` folder.
 3. Ensure the database user you've used for BookStack has limited permissions for only accessing the database used for BookStack data.
 4. Check that you've set the `APP_URL` option in your `.env` file so that system generated URLs cannot be manipulated.
 5. Within BookStack, go through the settings to ensure registration and public access settings are as you expect.
 6. Review the user roles in the settings area.
 7. Read the below to further understand the security for specific functionality and features.
-

Multi-Factor Authentication

Any user can enable multi-factor authentication (MFA) on their account. Upon login they would then need to use an extra proof of identity to gain access. BookStack currently supports the following mechanisms:

- TOTP (Time-based One-Time Passwords)
 - Labelled as "Mobile App" (Google/Microsoft Authenticator etc...).
 - Uses a SHA1 algorithm internally (Greater algorithms have poor cross-app compatibility).
- Backup Codes
 - These are a list of 16 one-time-use codes.
 - Users will be warned once they have less than 5 codes remaining.

Secrets and values for these options are stored encrypted within the database.

Where required, MFA can be forced upon users via their roles. This can be found via a "Requires Multi-Factor Authentication" checkbox seen when editing a role. If a user does not already have an MFA method configured, they will be forced to set one up upon next login.

Securing Images

By default, Images are stored in a way which is publicly accessible which ensures performance while using BookStack. Listed below are a few options for increasing image security.

Alternative Storage Options

Review our [file upload storage options](#) documentation for image storage options that add additional access or permission control to image requests.

Complex Urls

In the settings area of BookStack you can find the option 'Enable higher security image uploads?'. Enabling this will add a 16 character random string to the name of image files to prevent easy guessing of URLs. This increases security without potential performance concerns.

Prevent Directory Indexes

It's important to ensure you disable 'directory indexes' to prevent unknown users from being able to navigate their way through your images. Here's the configuration for NGINX & Apache if your server allows directory indexes:

NGINX

```
# By default indexes are disabled on Nginx but if you have them enabled
# add this to your BookStack server block
location /uploads {
    autoindex off;
}
```

Apache

```
# Add this to your Apache BookStack virtual host if Indexes are enabled.
# If .htaccess file are enabled then the below should already be active.
<Location "/uploads">
    Options -Indexes
</Location>
```

You can test that indexes are disabled by attempting to navigate to

`<your_bookstack_url>/uploads/images` in the browser after having uploaded at least one image. You should not see a list of files but instead a BookStack "Not Found" page.

Attachments

Attachments, if not using Amazon S3, are stored in the `storage/uploads` directory. By default, unlike images, these are stored behind the application authentication layer so access depends on permissions you have set up at a role level and page level.

If you are using Amazon S3 for file storage then access will depend on your S3 permission settings. Unlike images, BookStack will not automatically attempt to make uploaded attachments publicly accessible.

Filesystem Permissions

It's usually a good idea to limit the file and folder access privileges for the user/group used to run the application PHP, which is typically the user/group for PHP or the running web-server processes. Limiting permissions can help avoid a range of potential vulnerability exploits.

Details on required [filesystem permissions can be found here](#).

User Passwords

User passwords, if not using an alternative authentication method, are stored in the database. These are hashed using the standard Laravel hashing methods which use the Bcrypt hashing algorithm.

JavaScript in Page Content

By default, JavaScript tags within page content is escaped when rendered. This can be turned off by setting `ALLOW_CONTENT_SCRIPTS=true` in your `.env` file. Note that even if you disable this escaping the WYSIWYG editor may still perform its own JavaScript escaping. This option will also alter the [CSP rules](#) set by BookStack.

This option disables some fundamental cross-site-scripting protections. Only use this option on secure instances, where only very trusted users can edit content

Web Crawler Control

The rules found in the `/robots.txt` file are automatically controlled via the "Allow public viewing?" setting. This can be overridden by setting `ALLOW_ROBOTS=false` or `ALLOW_ROBOTS=true` in your `.env` file. If you'd like to customise the rules this can be done via theme overrides.

Secure Cookies

BookStack uses cookies to track sessions, remember logins and for XSRF protection. When using HTTPS you may want to ensure that cookies are only sent back to the browser if the connection is over HTTPS. If you have set a `https APP_URL` option in your `.env` this will be enabled automatically but it can also be forced on by setting `SESSION_SECURE_COOKIE=true` in your `.env` file.

Host Iframe Control

By default BookStack will only allow itself to be embedded within iframes on the same domain as you're hosting on. This is done through a [CSP: frame-ancestors](#) header. You can add additional trusted hosts by setting a `ALLOWED_IFRAME_HOSTS` option in your `.env` file like the example below:

```
# Adding a single host
ALLOWED_IFRAME_HOSTS="https://example.com"

# Multiple hosts can be separated with a space
ALLOWED_IFRAME_HOSTS="https://a.example.com https://b.example.com"
```

Note: when this option is used, all cookies will be served with `SameSite=None` ([info](#)) set so that a user session can persist within the iframe.

Iframe Source Control

By default BookStack will only allow certain other hosts to be used as `src` values for embedded iframe/frame content within the application. This is done through a [CSP: frame-src](#) header. You can configure the list of trusted sources by setting a `ALLOWED_IFRAME_SOURCES` option in your `.env` file like the examples below:

```
# Adding a single host
ALLOWED_IFRAME_SOURCES="https://example.com"

# Multiple hosts can be separated with a space
ALLOWED_IFRAME_SOURCES="https://a.example.com https://b.example.com"

# Allow all sources
# This opens vulnerability risk and should only be done in secure & trusted environments.
ALLOWED_IFRAME_SOURCES="*"
```

By default this option is configured as follows:

```
ALLOWED_IFRAME_SOURCES="https://*.draw.io https://*.youtube.com https://*.youtube-nocookie.com  
https://*.vimeo.com"
```

Note: The source of 'self' will always be automatically added to this CSP rule. In addition, the host used for the diagrams.net integration (If enabled) will be automatically appended to the lists of hosts.

Failed Access Logging

An option is available to log failed login events to a log file which is useful to identify users having trouble logging in, track malicious login attempts or to use with tools such as Fail2Ban. This works with login attempts using the default email & password login mechanism or attempts via LDAP login. Failed attempts are **not logged** for "one-click" social or SAML2 options.

To enable this you simply need to define the `LOG_FAILED_LOGIN_MESSAGE` option in your `.env` file like so:

```
LOG_FAILED_LOGIN_MESSAGE="Failed login for %u"
```

The optional "%u" element of the message will be replaced with the username or email provided in the login attempt when the message is logged. By default messages will be logged via the `php error_log` function which, in most cases, will log to your webserver error log files.

Untrusted Server Side Requests

Some features, such as the PDF exporting, have the option to make http calls to external user-defined locations to do things such as load images or styles. This is disabled by default but can be enabled if desired. This is required for using WKHTMLtoPDF as your PDF export renderer. This should only be enabled in BookStack environments where BookStack users and viewers are fully trusted.

To enable untrusted server side requests, you need to define the `ALLOW_UNTRUSTED_SERVER_FETCHING` option in your `.env` file like so:

```
ALLOW_UNTRUSTED_SERVER_FETCHING=true
```

Note that this is not connected to the `ALLOWED_SSR_HOSTS` setting at all.

`ALLOW_UNTRUSTED_SERVER_FETCHING` is used for typically "untrusted" scenarios/functionality whereas `ALLOWED_SSR_HOSTS` is used for typically "trusted" user scenarios/functionality (where admin-level

permissions are required).

Server Side Request Allow List

There is some functionality in BookStack, like webhooks for example, that is typically only accessible by trusted (admin-level) users. In some cases, those admin users may not be trusted to the level where they can freely make server side requests. In such scenarios, you can define a `ALLOWED_SSR_HOSTS` option in your `.env` file to limit the hosts that can be called for such typically trusted functionality:

```
# Allow all hosts (Default)
ALLOWED_SSR_HOSTS="*"

# Allow a single host
ALLOWED_SSR_HOSTS="https://example.com"

# Allow multiple hosts via space-separated entries
ALLOWED_SSR_HOSTS="https://example.com https://example.org"

# Use '*' as wildcards and/or define required paths
ALLOWED_SSR_HOSTS="https://*.example.com https://example.org/bookstack/"
```

Values will be compared prefix-matched, case-insensitive, against called SSR urls.

Note that this is not connected to the `ALLOW_UNTRUSTED_SERVER_FETCHING` setting at all. `ALLOW_UNTRUSTED_SERVER_FETCHING` is used for typically "untrusted" scenarios/functionality whereas `ALLOWED_SSR_HOSTS` is used for typically "trusted" user scenarios/functionality (where admin-level permissions are required).

Content Security Policy (CSP)

BookStack serves responses with a CSP header to increase protection against malicious content. This is especially important in a system such as BookStack where users can create a variety of HTML content, especially so if you allow untrusted users to create content in your instance. The CSP rules set by BookStack are as follows:

- `frame-ancestors 'self'`
 - Restricts what websites can embed BookStack pages via iframes.
 - See the "[Host Iframe Control](#)" section above for details on expanding this rule to other hosts.

- `frame-source 'self' https://*.diagrams.net https://*.draw.io https://*.youtube.com https://*.youtube-nocookie.com https://*.vimeo.com https://embed.diagrams.net`
 - Restricts what sources are allowed to load for frames/iframes.
 - Can be configured via a `ALLOWED_IFRAME_SOURCES` .env option.
 - May be different depending on other configuration set.
- `script-src http: https: 'nonce-abc123' 'strict-dynamic'`
 - Restricts what scripts can be ran on a BookStack-served page.
 - Will not be set if the `ALLOW_CONTENT_SCRIPTS` .env option is active.
 - The nonce value used is randomly generated upon each request. It is automatically applied to any "Custom HTML Head Content" scripts.
- `object-src 'self'`
 - Restricts which embeddable content can be loaded onto a BookStack-served page.
 - Will not be set if the `ALLOW_CONTENT_SCRIPTS` .env option is active.
- `base-uri 'self'`
 - Restricts what `<base>` tags can be added to a BookStack-served page.

If needed you should be able to set additional CSP headers via your webserver. If there's a clash with an existing BookStack CSP header then browsers will generally favour the most restrictive policy.

MySQL SSL Connection

If your BookStack database is not on the same host as your web server, you may want to ensure the connection is encrypted using SSL between these systems. Assuming SSL is configured correctly on your MySQL server, you can enable this by defining the `MYSQL_ATTR_SSL_CA` option in your `.env` file like so:

```
# Path to Certificate Authority (CA) certificate file for your MySQL instance.
# When this option is used host name identity verification will be performed
# which checks the hostname, used by the client, against names within the
# certificate itself (Common Name or Subject Alternative Name).
MYSQL_ATTR_SSL_CA="/path/to/ca.pem"
```

Using BookStack Content Externally

In some scenarios you may use BookStack user-provided content externally (Accessed via the database or API). Such content is not guaranteed to be safe so keep security in mind when dealing with such content. In some cases, the system will apply some filtering to content in an attempt to prevent certain vulnerabilities, but this is not assured to be a bullet-proof defence.

Within its own interfaces, unless disabled, BookStack makes use of Content Security Policy (CSP) rules to heavily negate cross-site scripting vulnerabilities from user content. If displaying user

content externally, it's advised you also use defences such as CSP or other techniques such as disabling of JavaScript entirely.

- #4
- cocobook-sync

2026-10-28 10:56:21 UTC
- cocobook-sync

2026-10-29 01:42:29 UTC