Core Concepts

Ghost Security

Ghost is committed to developing secure, reliable products utilising all modern security best practices and processes.

The Ghost team is made up of full time staff employed by the Ghost Foundation as well as volunteer open source contributors and security experts. We do both consultation and penetration testing of our software and infrastructure with external security researchers and agencies.

We take security seriously at Ghost and welcome any peer review of our **open source codebase** to help ensure that it remains secure.

Security features

Device verification

All staff user login sessions from a new or unrecognized device must be verified with a code sent to the user's registered email address.

Email 2FA

Ghost can be configured to send two-factor authentication codes by email on all staff user logins.

Brute force protection

User login attempts and password reset requests are all limited to 5 per hour per IP address.

Automatic SSL

Ghost's CLI tool automatically configures SSL certificates for all new Ghost installs with Let's Encrypt by default.

Password hashing

Ghost follows **OWASP authentication standards** with all passwords hashed and salted properly using `bcrypt` to ensure password integrity.

Encoded tokens everywhere

All user invitation and password reset tokens are base64 encoded with serverside secret. All tokens are always single use and always expire.

SQLi prevention

Ghost uses **Bookshelf** ORM + **Knex** query builder and does not generate *any* of its own raw SQL queries. Ghost has no interpolation of variables directly to SQL strings.

Data validation and serialisation

Ghost performs strong serialisation and validation on all data that goes into the database, as well as automated symlink protection on all uploaded files.

XSS prevention

Ghost uses safe/escaped strings used everywhere, including and especially in all custom Handlebars helpers used in **Ghost Themes**

Standardised permissions

Ghost-CLI does not run as `root` and automatically configures all server directory permissions correctly according to **OWASP Standards**.

Dependency management

All Ghost dependencies are continually scanned using a combination of automated GitHub tooling and `yarn audit` to ensure their integrity.

Reporting vulnerabilities

Potential security vulnerabilities can be reported directly to us at `security@ghost.org`. The Ghost Security Team communicates privately and works in a secured, isolated repository for tracking, testing, and resolving security-related issues.

Responsible disclosure

The Ghost Security team is committed to working with security researchers to verify, reproduce and respond to legitimate reported vulnerabilities.

- Provide details of the vulnerability, including information needed to reproduce and validate the vulnerability and a Proof of Concept
- Make a good faith effort to avoid privacy violations, destruction and modification of data on live sites
- Give reasonable time to correct the issue before making any information public

Security issues always take precedence over bug fixes and feature work. We can and do mark releases as “urgent” if they contain serious security fixes.

We will publicly acknowledge any report that results in a security commit to **https://github.com/TryGhost/Ghost**

Issue triage

We’re always interested in hearing about any reproducible vulnerability that affects the security of Ghost users, including...

- Remote Code Execution (RCE)
- SQL Injection (SQLi)
- Server Side Request Forgery (SSRF)
- Cross Site Request Forgery (CSRF)
- Cross Site Scripting (XSS) but please read on before reporting XSS...

However, we're generally *not* interested in...

- **Privilege escalation** as result of trusted users publishing arbitrary JavaScript[1]
- HTTP sniffing or HTTP tampering exploits
- Open API endpoints serving public data
- Ghost version number disclosure
- Brute force, DoS, DDoS, phishing, text injection, or social engineering attacks.
- Output from automated scans
- Clickjacking with minimal security implications
- Missing DMARC records

Privilege escalation attacks

Ghost is a content management system and all users are considered to be privileged/trusted. A user can only obtain an account and start creating content after they have been invited by the site owner or similar administrator-level user.

A basic feature of Ghost as a CMS is to allow content creators to make use of scripts, SVGs, embedded content & other file uploads that are required for the content to display as intended. Because of this there will always be the possibility of "XSS" attacks, albeit only from users that have been trusted to build the site's content.

Ghost's admin application does a lot to ensure that unknown scripts are not run within the the admin application itself, however that only protects one side of a Ghost site. If the front-end (the rendered site that anonymous visitors see) shares the same domain as the admin application then browsers do not offer sufficient protections to prevent successful XSS attacks by trusted users.

If you are concerned that trusted users you invite to create your site will act maliciously the best advice is to split your front-end and admin area onto different domains (e.g. `https://mysite.com` and `https://admin.mysite.com/ghost/`). This way browsers offer greater built-in protection because credentials cannot be read across domains. Even in this case it should be understood that you are giving invited users completely free reign in content creation so absolute security guarantees do not exist.

Anyone concerned about the security of their Ghost install should read our **hardening guide**.

We take any attack vector where an untrusted user is able to inject malicious content very seriously and welcome any and all reports.

How reports are handled

If you report a vulnerability to us through the **security@ghost.org** mailing list, we will:

- Acknowledge your email within a week
- Investigate and let you know our findings within two weeks
- Ensure any critical issues are resolved within a month
- Ensure any low-priority issues are resolved within three months
- Credit any open source commits to you
- Let you know when we have released fixes for issues you report

Suggest edits Raise issue

Newsletters Overview


Powered by mintlify