26 things to check on a self-hosted ERPNext

Each item says why it matters, how to check it and the setting or command. Tick the box when you have checked it on your own site. The page at mith.tech/tools/erpnext-hardening-checklist keeps your marks in your browser and shows the commands for Docker.

Who can connect · 4 items

- [ ] HTTPS everywhere, with a certificate that renews itself (look first)
- [ ] HSTS is sent, and you know what it commits you to
- [ ] Only 80, 443 and SSH answer from outside (look first)
- [ ] MariaDB listens locally and has no example password (look first)

What the site trusts · 3 items

- [ ] Frappe knows its own address: host_name is set
- [ ] Cross-site access is off, or limited to named origins
- [ ] A site-wide request budget is set, or deliberately not

Signing in · 3 items

- [ ] Two-factor sign-in is on for everyone who uses the desk
- [ ] The password strength rule is on, and rotation is a decision
- [ ] Failed-login lock-out and the idle timeout are set on purpose

Accounts and keys · 4 items

- [ ] Administrator has its own strong password and is not used daily (look first)
- [ ] Few people hold System Manager, and leavers are disabled
- [ ] Each API key belongs to one integration, with only the roles it needs
- [ ] Guests cannot upload files unless a web form needs it

Backups and recovery · 5 items

- [ ] Backups run on a schedule, and they include files (look first)
- [ ] A copy of every backup leaves the server (look first)
- [ ] Backups are encrypted, and the key is kept apart from them
- [ ] You have restored a backup somewhere else, and timed it (look first)
- [ ] The site's encryption_key is saved apart from the database backups (look first)

The server and its files · 4 items

- [ ] Files that hold secrets are readable only by the bench user
- [ ] Developer mode is off, and Server Scripts only if you use them
- [ ] The scheduler is running
- [ ] Logs are kept long enough to check, and rotate before the disk fills

Staying current · 3 items

- [ ] You run a Frappe version that still gets fixes (look first)
- [ ] Patch releases are applied on a calendar you can state
- [ ] The server's own software gets security updates

Look first: an outsider can use the gap with no account, or the damage cannot be repaired afterwards. Commands assume you run them from the bench folder. In Docker, replace bench with docker compose exec backend bench. show-config prints every key, passwords included: always pipe it through grep.

Who can connect

The front door: HTTPS, open ports, the database.

☐ HTTPS everywhere, with a certificate that renews itself

LOOK FIRST

Without HTTPS, passwords, session cookies and API secrets cross the network in the clear. An expired certificate does the opposite damage: browsers and webhook senders stop talking to the site until someone notices.

CHECK

FROM YOUR COMPUTER
curl -sI http://erp.example.com | head -n 3
ON THE SERVER
sudo certbot certificates
ON THE SERVER
sudo crontab -l | grep -i certbot
FROM YOUR COMPUTER
nmap --script ssl-enum-ciphers -p 443 erp.example.com

Good: The first shows a 301 to https://. The second shows an expiry date in the future. The third shows a certbot renew line. The fourth lists TLS 1.2 and 1.3 and nothing older.

SET · ssl_certificate and ssl_certificate_key in site_config.json, written by bench

ON THE SERVER
bench config dns_multitenant on
bench setup lets-encrypt refuses to run without it.
ON THE SERVER
sudo -H bench setup lets-encrypt erp.example.com
The domain must already point at the server, with port 80 open.

Sources: bench: bench/config/templates/nginx.conf · bench: bench/config/lets_encrypt.py · bench: bench/commands/config.py · Frappe docs: Let's Encrypt SSL setup · +3 more

☐ HSTS is sent, and you know what it commits you to

Bench and the Docker image both send a two-year HSTS header that covers every subdomain and carries the preload flag. After one visit, a browser refuses plain HTTP to your domain and to every subdomain of it. That is the aim. It bites when a subdomain has no certificate.

CHECK

FROM YOUR COMPUTER
curl -sI https://erp.example.com | grep -i strict-transport

List the subdomains of your domain in DNS and open each one over https://.

Good: The header is there, and every subdomain opens over HTTPS with no warning.

SET · No Frappe setting: the header comes from the nginx file

To change the header, set your own in the proxy in front of Frappe. bench setup nginx rewrites config/nginx.conf, so edits there are lost.

Putting the domain on the browsers' preload list (hstspreload.org) is a separate step. Do not take it until every subdomain is on HTTPS for good: removal takes months.

Sources: bench: bench/config/templates/nginx.conf · frappe_docker: resources/core/nginx/security_headers.conf · OWASP: HTTP Strict Transport Security Cheat Sheet · hstspreload.org

☐ Only 80, 443 and SSH answer from outside

LOOK FIRST

In bench's own files gunicorn (8000) and Redis (11000, 13000) bind to 127.0.0.1. The real-time service (9000) listens on every interface, and MariaDB (3306) listens wherever its config says. A firewall is what keeps all of them off the internet. In Docker, a published port skips ufw.

CHECK

FROM YOUR COMPUTER
nmap -Pn -p 22,80,443,3306,6379,8000,9000,11000,13000 erp.example.com
ON THE SERVER
sudo ss -ltnp

Good: From outside, only 22, 80 and 443 are open. On the server, anything listening on 0.0.0.0 or :: besides those has a reason you can state.

SET · A firewall, not a Frappe setting

ON THE SERVER
sudo bench setup firewall
Denies everything but 80, 443 and the SSH port: ufw on Ubuntu and Debian, firewalld on CentOS.

Or close the same ports in your cloud provider's firewall, which sits outside the server and which Docker cannot bypass.

Sources: bench: bench/playbooks/roles/bench/tasks/setup_firewall.yml · bench: bench/commands/setup.py · bench: bench/config/templates/redis_queue.conf · bench: bench/config/templates/supervisor.conf · +4 more

MariaDB listens locally and has no example password

LOOK FIRST

MariaDB holds every ledger entry. Frappe gives each site its own database user, with rights on that site's database only and, typically, only from the same machine. The usual breaks: the server listens on the network, or a Docker install keeps the example root password, 123.

CHECK

ON THE SERVER
sudo ss -ltn | grep 3306
ON THE SERVER
sudo mariadb -e "SELECT user, host FROM mysql.user"
ON THE SERVER
grep -c '^DB_PASSWORD=123$' .env
Docker installs only, from the folder that holds .env.

Good: 3306 is on 127.0.0.1, or on a private address if the database has its own machine. No user has host % unless it does. In Docker the count is 0.

SET · bind-address in the MariaDB config; DB_PASSWORD in frappe_docker's .env

In the MariaDB config set bind-address = 127.0.0.1 and restart MariaDB. The file's name depends on the distribution.

For a database on another machine, create the site with --mariadb-user-host-login-scope set to that machine's address, not %, and keep 3306 closed to everything else.

In Docker, set DB_PASSWORD in .env before the first start.

Sources: MariaDB: configuring for remote client access · frappe/database/mariadb/setup_db.py · frappe/commands/site.py · frappe_docker: example.env · +2 more

What the site trusts

Its own address, other sites, and request limits.

☐ Frappe knows its own address: host_name is set

Frappe has no allowed-hosts list; we found none in the source. Without host_name, some links are built from the request's Host header. In July 2026 a high-severity advisory (CVE-2026-47194, fixed in v15.108.0 and v16.18.3) showed login links sent to a forged host. Setting host_name pins the address.

CHECK
IN THE BENCH
bench --site erp.example.com execute frappe.utils.get_url
Good: It prints your public address, starting https://. Not http://, not a bench port.

SET · host_name in site_config.json
IN THE BENCH
bench --site erp.example.com set-config host_name https://erp.example.com

Sources: frappe/utils/data.py · frappe/www/login.py · frappe/core/doctype/user/user.py · frappe/utils/data.py · +3 more

☐ Cross-site access is off, or limited to named origins

With allow_cors set to "*", Frappe repeats any Origin back and says credentials are allowed. Frappe's own docs call the wildcard not recommended for production. The session cookie is SameSite=Lax, which limits what a hostile page can do, so this is a setting to remove, not an open door. Most sites need no cross-origin access at all.

CHECK
FROM YOUR COMPUTER
curl -sI -H "Origin: https://evil.example" https://erp.example.com/api/method/ping | grep -i access-control
IN THE BENCH
bench --site erp.example.com show-config | grep allow_cors
show-config prints every key, passwords included. Always filter it.
Good: No Access-Control line from the first. No allow_cors line from the second, or only your own origins.

SET · allow_cors in site_config.json
IN THE BENCH
bench --site erp.example.com set-config allow_cors None
Removes the key.
IN THE BENCH
bench --site erp.example.com set-config allow_cors '["https://portal.example.com"]' --parse
Or a list of your own origins.

Sources: frappe/app.py · frappe/auth.py · frappe/__init__.py · frappe/commands/utils.py · +4 more

☐ A site-wide request budget is set, or deliberately not

Frappe can add up the time all requests take and answer 429 to everyone once a window's budget is gone. That caps what a flood can cost the server, but one noisy client can also spend everyone's budget, and it is not a login lock-out. Set too low it blocks your own staff, so watch the headers first. Bench can also cap concurrent connections per site name in nginx.

CHECK
FROM YOUR COMPUTER
curl -sI https://erp.example.com/api/method/ping | grep -i x-ratelimit
IN THE BENCH
bench --site erp.example.com show-config | grep rate_limit
Good: Either X-RateLimit headers come back and rate_limit.limit and rate_limit.window are set, or you decided against a budget and can say why.

SET · rate_limit in site_config.json; allow_rate_limiting in common_site_config.json
IN THE BENCH
bench --site erp.example.com set-config rate_limit "{'limit': 600, 'window': 3600}" --parse
600 and 3600 are the numbers in Frappe's docs: seconds of request time per window of seconds. Start from what your site uses.
ON THE SERVER
bench set-config -g allow_rate_limiting true
Then bench setup nginx and reload nginx: it adds a cap of 8 concurrent connections per site name.

Sources: frappe/rate_limiter.py · Frappe docs: rate limiting · bench: bench/config/nginx.py · bench: bench/config/templates/nginx.conf · +2 more

Signing in

The second factor, passwords and lock-out.

☐ Two-factor sign-in is on for everyone who uses the desk

OWASP says a second factor is the main defence against most password attacks: brute force, credential stuffing, spraying. Turning on Enable Two Factor Auth in System Settings also ticks Two Factor Authentication on the role All, so it covers every user, Administrator included.

CHECK

IN THE BENCH

bench --site erp.example.com execute frappe.db.get_single_value --args "['System Settings','enable_two_factor_auth']"

Prints 1 when it is on and nothing when it is off.

Sign in as an ordinary user in a private window.

Good: The command prints 1 and the ordinary user is asked for a code.

SET · enable_two_factor_auth and two_factor_method in System Settings

System Settings → Two Factor Authentication: tick Enable Two Factor Auth and keep the method OTP App.

Leave "Bypass Two Factor Auth for users who login from restricted IP Address" unticked unless a fixed-address rule is part of your plan.

To limit it to some roles, untick Two Factor Authentication on the role All and tick it on those roles.

Note: Frappe's ERPNext docs page says to run bench set-config enable_two_factor_auth true first. The code at both pinned versions reads only the System Settings field, so ticking it is enough. API keys do not ask for a code either.

Sources: frappe/core/doctype/system_settings/system_settings.py · frappe/twofactor.py · frappe/auth.py · frappe/auth.py · +4 more

☐ The password strength rule is on, and rotation is a decision

Frappe scores a new password with zxcvbn and refuses one under the minimum score. It is on by default at 2, which resists online guessing but not a stolen database; 3 adds some protection against offline attacks. Forced rotation is your call: NIST says not to require periodic changes, unless a password is known to be compromised, and not to impose composition rules.

CHECK

IN THE BENCH

bench --site erp.example.com execute frappe.db.get_single_value --args "['System Settings','minimum_password_score']"

IN THE BENCH

bench --site erp.example.com execute frappe.db.get_single_value --args "['System Settings','force_user_to_reset_password']"

Prints nothing when it is empty.

Good: The score is 3 or 4 (our suggestion; Frappe's default is 2). Reset-after-days is empty unless a rule of yours needs it.

SET · enable_password_policy, minimum_password_score and force_user_to_reset_password in System Settings

System Settings → Password: keep Enable Password Policy ticked and raise Minimum Password Score.

Administrator and Guest are never forced to reset; change Administrator's password by hand (see Accounts and keys).

Sources: frappe/core/doctype/system_settings/system_settings.json · frappe/core/doctype/system_settings/system_settings.json · frappe/core/doctype/user/user.py · frappe/utils/password.py · +2 more

☐ Failed-login lock-out and the idle timeout are set on purpose

Frappe counts failed sign-ins per account and per IP address and locks for a while after too many: 10 tries, then 60 seconds, by default. Zero tries switches lock-out off. A new site's idle timeout is 170 hours, about a week. Bench installs fail2ban, but its one jail watches for proxy abuse and scanner paths, not sign-ins.

CHECK

IN THE BENCH

bench --site erp.example.com execute frappe.db.get_single_value --args "['System Settings','allow_consecutive_login_attempts']"

IN THE BENCH

bench --site erp.example.com execute frappe.db.get_single_value --args "['System Settings','session_expiry']"

Good: Tries is above 0, the lock time is one you chose, and the idle timeout suits where your people sit.

SET · allow_consecutive_login_attempts, allow_login_after_fail and session_expiry in System Settings

System Settings → Brute Force Security: the number of tries and the lock time in seconds.

System Settings → Session Expiry (idle timeout), written hours:minutes, such as 08:00.

Allow only one session per user is off by default, and mobile devices are exempt from it.

Sources: frappe/core/doctype/system_settings/system_settings.json · frappe/auth.py · frappe/sessions.py · bench: bench/config/production_setup.py · +2 more

Accounts and keys

Administrator, System Managers, API keys, uploads.

☐ Administrator has its own strong password and is not used daily

LOOK FIRST

Administrator exists on every site, and it is the account Frappe never forces to change its password. Frappe has no built-in default: the password is whatever was typed at install, or put in admin_password in a config file. A config file that still holds it is a plain-text copy.

CHECK

IN THE BENCH

grep -l admin_password sites/common_site_config.json sites/erp.example.com/site_config.json

Good: No file is listed. Day-to-day work happens under named users, and you can say who knows the Administrator password.

SET · The Administrator password; admin_password in config

IN THE BENCH

bench --site erp.example.com set-admin-password --logout-all-sessions

Asks for the password, so it stays out of your shell history.

IN THE BENCH

bench --site erp.example.com set-config admin_password None

IN THE BENCH

bench set-config -g admin_password None

Sources: frappe/commands/site.py · frappe/utils/install.py · frappe/auth.py · frappe/__init__.py · +2 more

☐ Few people hold System Manager, and leavers are disabled

A System Manager can create API keys for any user and download backups. A person who has left can still sign in until their account is disabled. Review who holds the role, and who is still enabled, on a calendar.

CHECK

User list → filter Role is System Manager and Enabled is Yes. Then sort the enabled users by Last Active.

Good: You can name every System Manager and each needs the role. No leaver is still enabled, and nobody has been unseen for months.

SET · Enabled on the User form

IN THE BENCH

bench --site erp.example.com disable-user person@example.com

Or untick Enabled on the User form.

Sources: frappe/core/doctype/user/user.py · frappe/utils/response.py · frappe/commands/site.py · frappe/core/doctype/user/user.json

☐ Each API key belongs to one integration, with only the roles it needs

An API key and secret sign in with no password and no 2FA code, so the key's user is the whole gate. Until v15.120.0 and v16.33.0 a user's IP allow-list was not applied to key requests (advisory GHSA-h9jq-7pgf-hw8q, October 2026). Generating keys again makes a new secret and the old one stops working.

CHECK

User list → filter API Key is set. For each user, name the integration and read their roles.

IN THE BENCH

bench version

Good: Every key maps to one named integration, with the fewest roles that work and an IP allow-list where its address is fixed. frappe is at v15.120.0 or later, or v16.33.0 or later.

SET · API Access and Restrict IP on the User form

One user per integration, with only the roles that integration needs.

User → Restrict IP: the addresses it calls from.

User → API Access → Generate Keys to rotate: copy the secret when it is shown.

Disable the user when the integration ends.

Sources: frappe/auth.py · frappe/auth.py · frappe/core/doctype/user/user.py · frappe/core/doctype/user/user.json · +2 more

☐ Guests cannot upload files unless a web form needs it

Allow Guests to Upload Files is off by default. If a public form needs it, narrow what can arrive: Frappe can limit guest uploads to named DocTypes, limit file extensions and cap the size. v16 can also restrict public uploads to System Managers.

CHECK

IN THE BENCH

bench --site erp.example.com execute frappe.db.get_single_value --args "['System Settings','allow_guests_to_upload_files']"

Good: It prints nothing (off), or you can name the web form that needs it and the DocType list and the extensions are filled in.

SET · allow_guests_to_upload_files, allowed_doctypes_for_guest_uploads, allowed_file_extensions and max_file_size in System Settings; in v16 also only_allow_system_managers_to_upload_public_files

System Settings → untick Allow Guests to Upload Files, or fill in Allowed Doctypes for Guest Uploads, Allowed File Extensions and Max File Size.

Sources: frappe/core/doctype/system_settings/system_settings.json · frappe/core/doctype/system_settings/system_settings.json · frappe/core/doctype/system_settings/system_settings.json · OWASP Top 10 (2021): A05 Security Misconfiguration

Backups and recovery

Copies that exist, leave the server and restore.

☐ Backups run on a schedule, and they include files

LOOK FIRST

bench init adds a cron job that backs up every site every six hours. It takes the database only: uploaded files need --with-files. Each run also deletes backup files older than 23 hours and the server keeps three sets, so what sits on it is about a day of history.

CHECK

ON THE SERVER

crontab -l | grep backup

Run as the user that owns the bench.

IN THE BENCH

ls -lh sites/erp.example.com/private/backups

Good: A cron line runs backup with --with-files, and the folder holds a fresh -database.sql.gz, -files.tar and -private-files.tar.

SET · The cron job; keep_backups_for_hours in site_config.json; Number of Backups in System Settings

ON THE SERVER

cd ~/frappe-bench && bench --verbose --site all backup --with-files >> logs/backup.log 2>&1

The line to put in crontab -e, every six hours, in place of the one bench wrote.

System Settings → Number of Backups (default 3) sets how many sets stay on the server.

Sources: bench: bench/bench.py · bench: bench/utils/system.py · frappe/commands/site.py · frappe/utils/backups.py · +5 more

☐ A copy of every backup leaves the server

LOOK FIRST

Backups on the server die with it, and the server throws them away within about a day. A dead disk, a ransomware run or a deleted instance takes both. Keep copies somewhere the server can add to but not delete from (our advice, not Frappe's).

CHECK

Open the place the copies go (a bucket, a storage box) and sort by date.

Good: The newest copy is from the last backup run, and older ones follow a retention rule you chose.

SET · S3 Backup Settings, Dropbox Settings or Google Drive in v15; the offsite_backups app in v16

v15: Integrations → S3 Backup Settings, Dropbox Settings or Google Drive. Each runs a daily job.

ON THE SERVER

bench get-app https://github.com/frappe/offsite_backups --branch version-16

v16 moved those three out of Frappe core into this app.

IN THE BENCH

bench --site erp.example.com install-app offsite_backups

Or copy sites/{site}/private/backups yourself: frappe_docker documents restic and a push_backup.py script for S3-compatible storage. Copy the archives, not the site_config copy (next item).

Sources: frappe/utils/backups.py · frappe/hooks.py · frappe/integrations/doctype/s3_backup_settings, frappe/integrations/doctype/dropbox_settings, frappe/integrations/doctype/google_drive · offsite_backups: offsite_backups/hooks.py · +2 more

☐ Backups are encrypted, and the key is kept apart from them

Encrypt Backups is off by default. When on, Frappe encrypts the database and the file archives with gpg, using a separate key it creates in site_config.json. The copy of site_config.json that goes into each backup is not encrypted, though its name ends in -enc. It holds the database password, the encryption_key and, once encryption has run, the backup key.

CHECK

IN THE BENCH

bench --site erp.example.com execute frappe.db.get_single_value --args "['System Settings','encrypt_backup']"

ON THE SERVER

gpg --version | head -n 1

IN THE BENCH

file sites/erp.example.com/private/backups/*

Docker: run it through sh -c so the * expands.

Good: encrypt_backup prints 1 and gpg is installed. The archives show as PGP encrypted data. The site_config copy shows as JSON text: that is the file to keep out of the off-box copy.

SET · encrypt_backup in System Settings; backup_encryption_key in site_config.json

System Settings → Backups → Encrypt Backups, with gpg installed.

Save the key now: Download Backups → Get Encryption Key (System Manager), or the backup_encryption_key line in site_config.json.

Keep the *_site_config_backup-enc.json files out of the off-box copy, or encrypt them with your own tool first.

To restore an encrypted backup on another site, copy backup_encryption_key into that site's site_config.json first, then run bench restore with no key option (the restore item has the full steps). bench restore reads the key from the site it restores into. A passphrase typed on the command line stays in your shell history.

Note: Frappe's how-to page names the restore flag --backup-encryption-key. The code at both pinned versions has no option of that name: it reads --encryption-key, and with no option it takes backup_encryption_key from the site it restores into, as the same page says a plain restore does. Use that, so the passphrase stays off the command line.

Sources: frappe/utils/backups.py · frappe/utils/backups.py · frappe/core/doctype/system_settings/system_settings.json · frappe/commands/site.py · +4 more

You have restored a backup somewhere else, and timed it

LOOK FIRST

A backup you have never restored is a guess. A restore finds the quiet failures: files missing from the archive, a key nobody saved, an encrypted archive with no passphrase. Do it on a scratch bench, never on the live site.

CHECK

ON THE SERVER

bench new-site restore-test.local

Asks for the MariaDB root and Administrator passwords.

Copy encryption_key (and backup_encryption_key, if the archives are encrypted) from the live site_config.json into restore-test.local's. bench restore reads the backup key from there, so no key goes on the command line.

ON THE SERVER

bench --site restore-test.local set-config mute_emails 1

Before the restore. It stops the email queue sending, so the copy sends no real email, including mail an action on it sends straight away. A Send Now click on an Email Queue row still sends.

ON THE SERVER

bench --site restore-test.local restore PATH/…-database.sql.gz --with-public-files PATH/…-files.tar --with-private-files PATH/…-private-files.tar

ON THE SERVER

bench --site restore-test.local scheduler disable

A second guard. A new site starts with the scheduler off and the restore keeps that setting, so it should already be off.

ON THE SERVER

bench --site restore-test.local execute frappe.utils.password.get_decrypted_password --args "['Email Account','NAME','password']" | wc -c

Prints a count, not the password. NAME is an Email Account that has a password.

Good: The site opens, a recent record and its attachment open, the last command prints a number above 1 (an 'Encryption key is invalid' error means the key is wrong), and you wrote down how long it took.

SET · No setting: a habit

Repeat it after every major upgrade, after any change to the backup tooling, and at least every quarter (our advice).

Sources: frappe/commands/site.py · frappe/commands/site.py · frappe/utils/password.py · frappe/commands/scheduler.py · +5 more

The site's encryption_key is saved apart from the database backups

LOOK FIRST

Password-type fields (API secrets, email account passwords, integration credentials) are stored encrypted with encryption_key from site_config.json. Frappe creates the key the first time one is saved. A database restored without it keeps the rows but cannot read them, and Frappe's own error tells you to copy the original site_config.json. Changing the key breaks every one of them.

CHECK

IN THE BENCH

grep -c '"encryption_key"' sites/erp.example.com/site_config.json

Prints a count, not the key.

Ask where a copy of the key is kept, and whether that place is different from where the database backups are.

Good: The count is 1, and a copy is in a password manager or secrets vault, not in the bucket that holds the database dumps. If the count is 0, nothing has been encrypted yet, but it will be.

SET · No setting: keep a copy of encryption_key

Read the value once as the bench user and store it in a password manager or secrets vault. Do not paste it into chat or a ticket.

Never regenerate it. If it is lost, every stored password (email accounts, integrations, API secrets) has to be entered again.

Sources: frappe/utils/password.py · frappe/utils/password.py · frappe/installer.py · Frappe docs: site_config.json · +2 more

The server and its files

Permissions, developer mode, the scheduler, logs.

☐ Files that hold secrets are readable only by the bench user

site_config.json holds the database password and the encryption keys in plain text; common_site_config.json holds the Redis addresses. Frappe never sets a file mode, so they get whatever the process umask gives, commonly readable by every local account. Backups in private/backups are the same.

CHECK

IN THE BENCH

stat -c '%a %U:%G %n' sites/common_site_config.json sites/erp.example.com/site_config.json

IN THE BENCH

stat -c '%a %n' sites/erp.example.com/private/backups

Good: Both config files show 600 or 640, owned by the bench user, with a group that nobody else shares.

SET · File modes, not a Frappe setting

ON THE SERVER

chmod 600 sites/common_site_config.json sites/erp.example.com/site_config.json

Run from the bench folder, as its owner.

Leave sites/{site}/private alone: nginx reads files there on Frappe's behalf, so a 700 mode breaks private downloads.

Sources: frappe/installer.py · frappe/installer.py · bench: bench/utils/system.py · bench: bench/config/templates/nginx.conf · +3 more

☐ Developer mode is off, and Server Scripts only if you use them

In developer mode Frappe allows changes to standard DocTypes and writes them to files; without it those changes are refused. A production site has no reason to allow that. Server Scripts run Python written in the desk, and they stay off unless server_script_enabled is set in common_site_config.json.

CHECK

IN THE BENCH

bench --site erp.example.com show-config | grep -E 'developer_mode|server_script_enabled'

Good: No developer_mode line, or 0. No server_script_enabled line unless you use Server Scripts.

SET · developer_mode in site_config.json; server_script_enabled in common_site_config.json

IN THE BENCH

bench --site erp.example.com set-config developer_mode 0

IN THE BENCH

bench set-config -g server_script_enabled 0

Sources: frappe/core/doctype/doctype/doctype.py · frappe/utils/safe_exec.py · Frappe docs: site_config.json · frappe/commands/utils.py · +2 more

☐ The scheduler is running

Scheduled jobs are part of the clean-up: expired sessions, old logs, the 2FA QR-code images and old downloadable backups. With the scheduler off, in maintenance mode or paused, none of that happens. Frappe also runs jobs only once a day on a site nobody has opened for 4 days.

CHECK

IN THE BENCH

bench --site erp.example.com scheduler status

Good: It says the scheduler is enabled.

SET · enable_scheduler in System Settings; pause_scheduler and maintenance_mode in site_config.json

IN THE BENCH

bench --site erp.example.com scheduler enable

IN THE BENCH

bench --site erp.example.com scheduler resume

Sources: frappe/commands/scheduler.py · frappe/utils/scheduler.py · frappe/hooks.py · frappe/core/doctype/system_settings/system_settings.json

☐ Logs are kept long enough to check, and rotate before the disk fills

Three layers keep logs, each with its own limit. The Activity Log, which holds sign-in history, is cleared after 90 days by default and the Error Log after 14. Frappe's own log files rotate at 100 KB, 20 files. Supervisor rotates its logs at 50 MB, 10 files; nginx's depend on your operating system.

CHECK

Log Settings → Logs to Clear: read the days against how long you need sign-in history.

ON THE SERVER

ls -lh logs/ && du -sh /var/log/nginx

Good: The days suit your needs, and no log is growing without a limit.

SET · Logs to Clear in Log Settings

Log Settings → change the days per log type. Export first anything you must keep longer than the default.

Sources: frappe/hooks.py · frappe/utils/logger.py · frappe/hooks.py · Supervisor docs: configuration · +1 more

Staying current

Supported versions and a patch calendar.

☐ You run a Frappe version that still gets fixes

LOOK FIRST

Frappe sends security fixes to each supported version separately, and none to a version that has ended. On Frappe's support page, version 14 ended on 31 January 2026; version 15 is planned to end at the end of 2027 and version 16 at the end of 2029.

CHECK

IN THE BENCH

bench version

Good: frappe and erpnext are on version 15 or 16, and you know which release.

SET · No setting: an upgrade plan

If you are on 14 or older, plan the move to 16: follow Frappe's migration guides and rehearse it on a restored copy first (see the restore item).

Sources: Frappe: supported versions · Frappe forum: ERPNext v14 has reached end of life · bench: bench/commands/update.py · frappe/commands/utils.py

☐ Patch releases are applied on a calendar you can state

Frappe tagged 17 v16 and 14 v15 releases in the 90 days to 14 October 2026. It published 59 security advisories for the framework in the 12 months to the same day, 15 of them high or critical. More advisories can mean more reports handled in the open, not a worse product. Fixes that are public are fixes you may be missing.

CHECK

IN THE BENCH

bench version

Compare with the latest tag at github.com/frappe/frappe/releases. Subscribe to github.com/frappe/frappe/releases.atom in a feed reader, or watch the repository for releases.

Good: You can say how old your release is, who applies updates, how often, and where they are tried first.

SET · No setting: a calendar. Disable System Update Notification in System Settings should stay unticked

IN THE BENCH

bench update

Backs up every site first (unless --no-backup), pulls, migrates, builds and restarts. Try it on a restored copy first.

Keep System Settings → Disable System Update Notification unticked so System Managers see new releases.

Sources: GitHub: frappe/frappe security advisories · GitHub: frappe/frappe releases · bench: bench/commands/update.py · frappe/utils/change_log.py · +1 more

☐ The server's own software gets security updates

Frappe's fixes cover Frappe. The operating system, nginx, MariaDB, Redis and Node are patched by their own vendors, and OWASP lists unpatched components as a risk category of its own. Ubuntu Server installs unattended-upgrades and applies security updates by default; other systems need it set up.

CHECK

ON THE SERVER

systemctl is-enabled unattended-upgrades

ON THE SERVER

apt list --upgradable 2>/dev/null | head -n 10

ON THE SERVER

ls /var/run/reboot-required

Good: The first says enabled, nothing security-related is waiting, and the third file is absent or a reboot is booked.

SET · unattended-upgrades on Ubuntu and Debian

Ubuntu: /etc/apt/apt.conf.d/20auto-upgrades turns it on and 50unattended-upgrades says what it installs and whether it reboots.

Sources: Ubuntu Server docs: automatic updates · OWASP Top 10 (2021): A06 Vulnerable and Outdated Components · MithTech: how to self-host on Hetzner

Your record

Where your answers go, for the next person.

Site

Checked by

Date

Frappe and ERPNext versions (bench version)

Where the backups go

Where the encryption_key copy is kept (not its value)

Date of the last restore test, and how long it took

Items marked not yet, and who owns each

What this is. A checklist with the source each item was read from, pinned to Frappe v16.52.0 and v15.123.0, bench v5.31.0 and frappe_docker v4.0.1 as read on 2026-10-14. It is not an audit, a scan or a score, and following it does not make a site secure. We read the code and the docs; not every command has been run on a live site. Settings and names change between releases: check your own version.