# Laravel + Server Hardening: Full Security Checklist (Deploy-Ready)

Use this as a step‑by‑step deployment guide and a recurring audit. Check off items as you complete them.

## 0) Scope & Assumptions

* Laravel **12.x** (works for 10+ with minor tweaks)
* PHP 8.2/8.3, Nginx or Apache, MySQL/PostgreSQL
* Linux server with systemd

## 1) Pre‑Deployment (Secrets & Config)

* Generate strong app key: php artisan key:generate
* **Do not commit** .env or secrets; use environment variables or a secret manager (e.g., SSM, Vault, Doppler).
* APP\_ENV=production, APP\_DEBUG=false, APP\_URL=https://your-domain.tld
* Configure trusted proxies if behind LB/CDN: App\Http\Middleware\TrustProxies
* Set timezone/locale correctly; avoid leaking debug info.
* Ensure queue/cron users have least privilege (non‑root).

**.env template (secure defaults):**

APP\_ENV=production  
APP\_DEBUG=false  
APP\_URL=https://example.com  
LOG\_CHANNEL=stack  
LOG\_LEVEL=warning  
SESSION\_DRIVER=file  
SESSION\_SECURE\_COOKIE=true  
SESSION\_HTTP\_ONLY=true  
SESSION\_SAME\_SITE=strict  
CACHE\_DRIVER=redis  
QUEUE\_CONNECTION=redis  
MAIL\_MAILER=smtp

## 2) Authentication & Authorization

* Use first‑party auth scaffolding (Breeze/Jetstream/Fortify) instead of custom.
* Enforce password hashing (bcrypt/argon2id). Keep password column length ≥ 255.
* Enable **2FA** for admin/privileged accounts.
* Implement **Policies/Gates** for all resource actions. Deny by default.
* Use **Sanctum** or OAuth2/Passport for APIs. Short token TTLs + rotation.
* Rate‑limit login/OTP endpoints (see §8).

## 3) Input Validation & Output Escaping

* Validate all requests with Form Requests (php artisan make:request), whitelist fields.
* Escape output with Blade {{ }}. Avoid {!! !!} unless sanitized.
* For rich text, sanitize via HTML Purifier on the server.
* Normalize encodings (UTF‑8) and reject invalid bytes.

## 4) File Upload Security (Web Shell Defense)

* Store uploads **outside** public/ (e.g., storage/app/uploads).
* Whitelist extensions & MIME; verify real image headers (exif\_imagetype).
* Rename to UUID; never use user‑supplied filename.
* **Disable PHP execution** in any web‑served upload directory.
* Size limits: max:2048 (or per business need).

**Route to serve private files:**

Route::get('/file/{path}', function ($path) {  
 return Storage::download('uploads/'.$path);  
})->where('path','.\*')->middleware('auth');

**Nginx (block PHP under /uploads):**

location ~\* /uploads/.\*\.(php|phtml|phar)$ { deny all; }  
location /uploads/ { autoindex off; try\_files $uri $uri/ =404; }

**Apache (.htaccess in /public/uploads):**

<FilesMatch "\.(php|phtml|phar)$">Deny from all</FilesMatch>  
RemoveHandler .php .phtml .phar  
RemoveType .php .phtml .phar

## 5) Database Security

* Use parameterized queries/Eloquent only; avoid string‑built SQL.
* Separate DB user with least privileges (no SUPER).
* Enforce TLS to DB where possible.
* Rotate credentials; no shared accounts.
* Migrate using php artisan migrate --force in CI/CD, not manually.
* Implement soft deletes & auditing on sensitive tables where relevant.

## 6) Sessions & Cookies

* SESSION\_SECURE\_COOKIE=true, SESSION\_HTTP\_ONLY=true, SESSION\_SAME\_SITE=strict
* Use cookie or file session driver for simple setups; Redis for scale.
* Regenerate session ID on login; use AuthenticateSession middleware for admins.

## 7) CORS & CSRF

* Enable CSRF middleware; include @csrf in all Blade forms.
* Use laravel-cors (built-in in 7+) with **strict** origins & allowed headers.
* Do not rely on \* for credentials; be explicit.

**config/cors.php (example):**

'paths' => ['api/\*', 'sanctum/csrf-cookie'],  
'allowed\_methods' => ['GET','POST','PUT','PATCH','DELETE'],  
'allowed\_origins' => ['https://app.example.com'],  
'allowed\_headers' => ['Content-Type','X-Requested-With','Authorization'],  
'supports\_credentials' => true,

## 8) Rate Limiting & Abuse Controls

* Throttle auth & write endpoints: throttle:login / throttle:60,1.
* Use IP + user key for fairness.
* Add captcha after repeated failures.

**routes/web.php:**

Route::middleware('throttle:10,1')->group(function(){  
 Route::post('/login', [AuthController::class, 'store']);  
});

**Custom limiter (RouteServiceProvider):**

RateLimiter::for('login', function (Request $request) {  
 return Limit::perMinute(10)->by($request->ip());  
});

## 9) Secure HTTP Headers & CSP

* Add X-Frame-Options: DENY, X-Content-Type-Options: nosniff, Referrer-Policy: no-referrer.
* Enforce HSTS once HTTPS is stable: Strict-Transport-Security: max-age=31536000; includeSubDomains; preload.
* Implement a **Content Security Policy** (start in Report‑Only, then enforce).

**Simple middleware example:**

$response->headers->set('X-Frame-Options','DENY');  
$response->headers->set('X-Content-Type-Options','nosniff');  
$response->headers->set('Referrer-Policy','no-referrer');  
$response->headers->set('Strict-Transport-Security','max-age=31536000; includeSubDomains; preload');

**Minimal CSP (tighten per app):**

$response->headers->set('Content-Security-Policy',  
 "default-src 'self'; img-src 'self' data: https:; script-src 'self'; style-src 'self' 'unsafe-inline'; connect-src 'self' https:; frame-ancestors 'none'; base-uri 'self'"  
);

## 10) Transport Security (TLS/HTTPS)

* Redirect HTTP→HTTPS at the edge.
* Use modern ciphers; enable OCSP stapling.
* Issue certificates via ACME (Let’s Encrypt) with auto‑renew.

**Nginx snippet:**

ssl\_protocols TLSv1.2 TLSv1.3;  
ssl\_prefer\_server\_ciphers on;  
ssl\_session\_cache shared:SSL:10m;  
add\_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload" always;

## 11) PHP & FPM Hardening

* expose\_php=Off in php.ini.
* Disable dangerous functions:

disable\_functions=exec,passthru,shell\_exec,system,proc\_open,popen,show\_source,parse\_ini\_file,pcntl\_exec,pcntl\_fork,dl

* Reasonable limits:

upload\_max\_filesize=2M  
post\_max\_size=2M  
max\_file\_uploads=5  
max\_execution\_time=30  
memory\_limit=256M

* Separate FPM pool user; no root. Restrict open\_basedir if applicable.

## 12) Web Server Hardening

**Nginx** - [ ] server\_tokens off; (hide version) - [ ] Deny hidden files: location ~ /\. → deny all; - [ ] Limit methods to expected ones.

**Apache** - [ ] ServerTokens Prod, ServerSignature Off - [ ] Deny access to .env, .git, backups.

**Block sensitive files (Nginx):**

location ~\* /(\.env|\.git|composer\.json|composer\.lock|\.env\.backup|\.htaccess)$ { deny all; }

## 13) Logging, Monitoring & Alerting

* Centralize logs (ELK, Loki, CloudWatch). Set LOG\_LEVEL=warning or higher.
* Capture auth events, permission denials, 4xx/5xx spikes.
* Enable health checks; export metrics (Prometheus/StatsD).
* Set up alerts for error rate, CPU, disk, queue failures, SSL expiry.

**Laravel Telescope (dev/staging only):** protect with auth/IP.

## 14) Dependency & Supply‑Chain Security

* composer audit in CI; fail builds on critical vulns.
* Pin versions; avoid abandoned/unmaintained packages.
* npm audit --audit-level=high for front‑end.
* Use private packagist/registry for internal packages.

## 15) CI/CD & Build Hygiene

* Build artifacts in CI; deploy read‑only code to servers.
* Run php artisan config:cache, route:cache, event:cache during build.
* No dev tools in production container/image.
* Secrets injected at deploy time via environment/secret store.

## 16) Backups & Key Management

* Nightly encrypted DB backups (AES‑256 or KMS), tested restore.
* Off‑site copies with retention policy.
* Rotate APP\_KEY only after re‑encryption planning; never share.
* Limit who can read backups.

## 17) Access Control & Least Privilege

* SSH via keys only; disable password auth; use MFA on bastion.
* Separate roles: admin vs deploy vs read‑only.
* Restrict outbound egress for app server if possible.

## 18) DDoS & Edge Security

* Put site behind CDN/WAF (Cloudflare/AWS WAF/Fastly) with basic rules.
* Rate‑limit at edge; JS challenge for abusive IPs.
* Block common CMS paths you don’t use (/wp-admin, etc.).

## 19) Web Shell & Malware Scanning

* Cron ClamAV/freshclam scans of web roots & uploads.
* Integrity check with tripwire/aide or CI artifact verification.
* Alert on unexpected new .php files in writable dirs.

## 20) Privacy & Data Protection

* Minimize PII; encrypt at rest where feasible (Laravel Crypt for fields).
* Mask sensitive values in logs.
* Data retention & deletion policies implemented.

## 21) Incident Response (IR) Runbook

**If compromise suspected:** 1. Isolate instance from network (security group / firewall). 2. Snapshot disks & memory for forensics. 3. Rotate all secrets (DB, tokens, APP\_KEY if needed with plan). 4. Invalidate sessions/tokens; force password resets. 5. Re‑deploy clean image from CI artifacts. 6. Conduct RCA; add detections & controls to prevent recurrence.

**Contacts & Escalation:** - On‑call engineer, Security lead, Cloud provider support, Legal/Compliance, Comms.

## 22) Quick Commands & Snippets

**Block env & dotfiles (Apache .htaccess):**

<FilesMatch "^(\.env|\.git|composer\.(json|lock)|artisan|\.htaccess)$">  
 Require all denied  
</FilesMatch>

**Deny dotfiles (Nginx):**

location ~ /\.[^/]+ { deny all; }

**Fail2ban (nginx 401/403 brute):** - Configure jail monitoring /var/log/nginx/access.log for repeated 401/403.

**Linux perms (Laravel):**

chown -R www-data:www-data storage bootstrap/cache  
find storage bootstrap/cache -type d -exec chmod 775 {} \;  
find storage bootstrap/cache -type f -exec chmod 664 {} \;

## 23) Optional Hardening Extras

* Content hashing & Subresource Integrity (SRI) for CDN assets.
* Encrypted columns (per‑field) for high‑sensitivity data.
* Device fingerprinting / risk‑based auth for admin.
* Security.txt and well‑defined vulnerability disclosure policy.

## 24) Audit Checklist (Printable)

**Application** - [ ] Auth/2FA enforced for admins - [ ] CSRF & CORS locked down - [ ] Rate limiting in place - [ ] File uploads locked & scanned - [ ] CSP & security headers set - [ ] Logs centralized & alerts configured

**Server** - [ ] TLS modern & HSTS - [ ] PHP hardened & updated - [ ] Web server denies sensitive files - [ ] Backups tested - [ ] WAF/CDN configured

**Process** - [ ] CI/CD with audits - [ ] Secrets rotation policy - [ ] IR runbook rehearsed

## 25) Appendices

### A) Example Security Middleware (headers)

class SecureHeaders  
{  
 public function handle($request, Closure $next)  
 {  
 $response = $next($request);  
 $response->headers->set('X-Frame-Options','DENY');  
 $response->headers->set('X-Content-Type-Options','nosniff');  
 $response->headers->set('Referrer-Policy','no-referrer');  
 $response->headers->set('Strict-Transport-Security','max-age=31536000; includeSubDomains; preload');  
 $response->headers->set('Cache-Control','no-store, no-cache, must-revalidate, private');  
 $response->headers->set('Content-Security-Policy',"default-src 'self'; img-src 'self' data: https:; script-src 'self'; style-src 'self' 'unsafe-inline'; connect-src 'self' https:; frame-ancestors 'none'; base-uri 'self'");  
 return $response;  
 }  
}

### B) Example Upload Security Middleware (web shell defense)

class FileUploadSecureMiddleware  
{  
 public function handle($request, Closure $next)  
 {  
 foreach ($request->allFiles() as $file) {  
 foreach ((array)$file as $f) { $this->check($f); }  
 }  
 return $next($request);  
 }  
 private function check($file)  
 {  
 if (!$file->isValid()) abort(400,'Invalid upload');  
 $bad = ['php','php3','php4','php5','phtml','phar','exe','sh','bat'];  
 $ext = strtolower($file->getClientOriginalExtension());  
 if (in\_array($ext,$bad)) abort(400,'Dangerous file type');  
 $allowed = ['image/jpeg','image/png','application/pdf'];  
 if (!in\_array($file->getMimeType(),$allowed)) abort(400,'Invalid type');  
 if (str\_starts\_with($file->getMimeType(),'image/')) {  
 if (!@exif\_imagetype($file->getPathname())) abort(400,'Fake image');  
 }  
 }  
}

### C) php.ini Hardening

expose\_php=Off  
upload\_max\_filesize=2M  
post\_max\_size=2M  
max\_file\_uploads=5  
max\_execution\_time=30  
memory\_limit=256M  
; Disable risky funcs  
disable\_functions=exec,passthru,shell\_exec,system,proc\_open,popen,show\_source,parse\_ini\_file,pcntl\_exec,pcntl\_fork,dl

**Pro tip:** Schedule a monthly 30‑minute “security pit‑stop” to run audits (composer audit, npm audit), review logs, rotate keys, and test restores.