

Coffee Beans

Checklist Sécurité Web.

130 points de contrôle pour réussir un audit ou un pentest : en-têtes HTTP, TLS, DNS & e-mail, XSS et injections, authentification, API, cloud, supply chain, RGPD.

Mode d'emploi

Comment l'utiliser

Cochez chaque point vérifié **en production**, pas seulement dans le code. Un auditeur teste ce qui est exposé.

Traitez d'abord tous les points **Critique** : ce sont ceux qui font échouer un audit ou qui sont exploités en premier.

Rejouez la checklist à chaque mise en prod majeure et au moins une fois par trimestre.

Priorités

- CRITIQUE** Bloquant : à corriger avant l'audit (43 points)
- HAUTE** Attendu par tout auditeur sérieux
- RECO** Défense en profondeur, maturité

01	En-têtes HTTP de sécurité	13
02	HTTPS & TLS	7
03	DNS, domaine & e-mail	14
04	Faillles XSS	9
05	Autres injections	9
06	Cookies, sessions & CSRF	6
07	Authentification	9
08	Contrôle d'accès	7
09	API, CORS & SSRF	9
10	Upload de fichiers	7
11	Secrets & configuration	8
12	Dépendances & supply chain	8
13	Infrastructure & cloud	10
14	Logs, détection & réponse	7
15	RGPD & conformité	7
A	Annexes : config prête à copier, commandes de vérif, outils	

Score final

/ 130

Critiques OK : / 43

En-têtes HTTP de sécurité

Premier point contrôlé par tout auditeur : 30 secondes avec securityheaders.com ou [Mozilla Observatory](https://mozilla.github.io/observatory/).

- ☐ **Strict-Transport-Security** CRITIQUE
`max-age=63072000; includeSubDomains; preload`. Puis inscription sur hstspreload.org.
- ☐ **Content-Security-Policy stricte** CRITIQUE
Nonces ou hashes + `'strict-dynamic'`, `object-src 'none'`, `base-uri 'none'`. Interdits : `'unsafe-inline'`, `'unsafe-eval'`, `*`.
- ☐ **CSP déployée d'abord en Report-Only** HAUTE
`Content-Security-Policy-Report-Only` + `report-to` pendant 1 à 2 semaines avant de passer en mode bloquant.
- ☐ **Anti-clickjacking** CRITIQUE
`frame-ancestors 'none'` dans la CSP + `X-Frame-Options: DENY` pour les vieux navigateurs.
- ☐ **X-Content-Type-Options** HAUTE
`nosniff` sur toutes les réponses, y compris les assets et l'API.
- ☐ **Referrer-Policy** HAUTE
`strict-origin-when-cross-origin` (ou `no-referrer` sur les pages sensibles).
- ☐ **Permissions-Policy** HAUTE
Couper ce qui ne sert pas : `camera=()`, `microphone=()`, `geolocation=()`, `payment=()`, `usb=()`.
- ☐ **Isolation cross-origin** RECO
`Cross-Origin-Opener-Policy: same-origin` et `Cross-Origin-Resource-Policy: same-origin`. `COEP: require-corp` seulement si besoin.
- ☐ **X-XSS-Protection retiré** HAUTE
Header obsolète : mettre `0` ou le supprimer (l'ancien filtre XSS a lui-même causé des failles).
- ☐ **Fuite de versions** HAUTE
Supprimer `Server`, `X-Powered-By`, `X-AspNet-Version`, ainsi que les versions dans les pages d'erreur.
- ☐ **Cache des pages authentifiées** HAUTE
`Cache-Control: no-store` sur les pages et réponses API contenant des données personnelles.
- ☐ **Déconnexion propre** RECO
`Clear-Site-Data: "cache", "cookies", "storage"` sur l'endpoint de logout.
- ☐ **Reporting** RECO
`Reporting-Endpoints` configuré pour collecter les violations CSP et les erreurs réseau.

HTTPS & TLS

___ / 7

Objectif : note A+ sur SSL Labs, testssl.sh sans alerte rouge.

- ☐ **TLS 1.2 minimum, TLS 1.3 activé** **CRITIQUE**
SSLv3, TLS 1.0 et 1.1 désactivés. Seules les suites ECDHE + AEAD (AES-GCM, ChaCha20) sont autorisées.
- ☐ **Redirection HTTP vers HTTPS** **CRITIQUE**
Une 301 unique vers la même URL en HTTPS, sur tous les hôtes (apex + www + sous-domaines).
- ☐ **Zéro contenu mixte** **CRITIQUE**
Aucune ressource http:// chargée. `upgrade-insecure-requests` dans la CSP en filet de sécurité.
- ☐ **Certificats renouvelés automatiquement** **HAUTE**
ACME / Let's Encrypt ou certificat managé, avec une alerte 21 jours avant expiration.
- ☐ **Chaîne complète** **HAUTE**
Certificats intermédiaires servis, clé $\geq$ RSA 2048 ou ECDSA P-256, SAN couvrant tous les hôtes.
- ☐ **Surveillance Certificate Transparency** **RECO**
Alerte en cas d'émission inattendue d'un certificat pour vos domaines (crt.sh, Cert Spotter).
- ☐ **Compression sur les réponses secrètes** **RECO**
Pas de compression HTTP sur les réponses mêlant secret et saisie utilisateur (BREACH).

DNS, domaine & e-mail

___ / 14

Le DNS est souvent oublié des devs, mais jamais des auditeurs ni des attaquants.

- ☐ **Verrou registrar + 2FA** **CRITIQUE**
`clientTransferProhibited` actif, 2FA sur le compte registrar, renouvellement auto, contacts à jour.
- ☐ **DNSSEC activé** **HAUTE**
Zone signée, enregistrement DS publié chez le registrar. Vérifiable sur dnsviz.net.
- ☐ **Enregistrement CAA** **HAUTE**
`0 issue "letsencrypt.org"` (uniquement vos CA) + `0 iodef "mailto:security@..."`.
- ☐ **Pas de sous-domaine orphelin** **CRITIQUE**
Aucun CNAME pointant vers une ressource supprimée (Netlify, Vercel, S3, Heroku, GitHub Pages). Risque : subdomain takeover.

- ☐ **SPF strict** CRITIQUE
Un seul enregistrement, au plus 10 lookups DNS, se terminant par `-all` (pas `~all` ni `+all`).
- ☐ **DKIM** CRITIQUE
Clés ≥ 2048 bits pour chaque service d'envoi (Google, Sweego, Brevo...), avec rotation.
- ☐ **DMARC en p=reject** CRITIQUE
`v=DMARC1; p=reject; rua=mailto:...; adkim=s; aspf=s`. Progression : none, puis quarantine, puis reject.
- ☐ **MTA-STS + TLS-RPT** HAUTE
Impose le TLS aux serveurs qui vous envoient du mail et fournit les rapports d'échec.
- ☐ **Domaines qui n'envoient pas de mail** HAUTE
`v=spf1 -all`, DMARC `p=reject`, MX nul `0`, pour empêcher leur usurpation.
- ☐ **Transfert de zone fermé** HAUTE
AXFR refusé depuis l'extérieur (`dig axfr @ns domaine` doit échouer).
- ☐ **Résilience DNS** RECO
Au moins 2 serveurs NS, idéalement chez 2 fournisseurs. TTL cohérents.
- ☐ **IP d'origine masquée** RECO
Le serveur d'origine n'accepte que le trafic du CDN/WAF. Pas de fuite via un vieil A record ou un en-tête d'e-mail.
- ☐ **Typosquatting** RECO
Surveillance des domaines proches (dnstwist) et rachat des variantes critiques.
- ☐ **security.txt** RECO
`/.well-known/security.txt` avec Contact, Expires et Policy (RFC 9116).

Faibles XSS

Cible n°1 des pentesteurs web. Chaque point d'entrée utilisateur est testé.

- ☐ **Échappement contextuel en sortie** CRITIQUE
Corps HTML, attribut, JS, URL et CSS ont chacun leur échappement. S'appuyer sur l'auto-escape du framework.
- ☐ **Aucun HTML brut non assaini** CRITIQUE
Bannir `innerHTML`, `dangerouslySetInnerHTML`, `v-html`, `set:html` et `{@html}` avec une donnée utilisateur, sauf passage par DOMPurify.

- ☐ **Schémas d'URL filtrés** CRITIQUE
- Bloquer `javascript:`, `data:` et `vbscript:` dans les `href` / `src` fournis par l'utilisateur. Liste blanche `http(s)/mailto`.
- ☐ **XSS DOM** HAUTE
- Pas de `eval`, `new Function`, `setTimeout(string)` ni `document.write`. Lectures de `location.hash` / `search` assainies.
- ☐ **JSON dans un `<script>`** HAUTE
- Échapper `<`, `>`, `&`, U+2028 et U+2029 lors de l'injection d'un état serveur dans la page.
- ☐ **Trusted Types** HAUTE
- `require-trusted-types-for 'script'` dans la CSP pour verrouiller les sinks DOM (Chromium).
- ☐ **Markdown & rich text** HAUTE
- Rendu côté serveur avec une liste blanche de balises et d'attributs. Les `on*=` et `style` sont retirés.
- ☐ **SVG et fichiers uploadés** HAUTE
- Pas de SVG utilisateur servi inline sur le domaine principal (il peut contenir du JS).
- ☐ **Tests automatisés** RECO
- Payloads de référence (`"><svg onload=alert(1)>`, polyglots) sur chaque champ, dans la CI ou via ZAP.

Autres injections

Toute donnée externe est hostile : `query`, `body`, `headers`, `cookies`, `fichiers`, `webhooks`.

- ☐ **SQL injection** CRITIQUE
- Requêtes paramétrées ou ORM partout. Aucune concaténation, y compris dans `ORDER BY` (liste blanche).
- ☐ **NoSQL injection** CRITIQUE
- Rejeter les opérateurs (`$ne`, `$gt`, `$where`) dans les entrées, typer strictement les champs.
- ☐ **Injection de commande OS** CRITIQUE
- Pas de `exec` / `system` avec une entrée utilisateur. Sinon `execFile` + arguments en tableau + liste blanche.
- ☐ **SSTI** HAUTE
- Jamais de template construit à partir d'une entrée utilisateur (Jinja, Twig, Handlebars, EJS...).
- ☐ **Path traversal** HAUTE
- Normaliser le chemin puis vérifier qu'il reste dans le dossier autorisé. Refuser `../` et `%2e%2e`.

- ☐ **XXE** HAUTE
- Entités externes et DTD désactivées dans tous les parseurs XML (SVG, DOCX et SAML inclus).
- ☐ **CRLF / header injection** HAUTE
- Retirer `\r\n` des valeurs placées dans les en-têtes, cookies, redirections et e-mails.
- ☐ **Prototype pollution (JS)** HAUTE
- Bloquer `__proto__`, `constructor` et `prototype` dans les merges profonds. Utiliser `Object.create(null)` ou `Map`.
- ☐ **Open redirect** RECO
- Paramètres `?next=` / `?redirect=` limités à des chemins relatifs ou à une liste blanche de domaines.



06 · A07 · AUTHENTICATION FAILURES

Cookies, sessions & CSRF

___ / 6

- ☐ **Flags des cookies de session** CRITIQUE
- `Secure; HttpOnly; SameSite=Lax` (ou `Strict`), préfixe `__Host-`, pas de `Domain` large.
- ☐ **Protection CSRF** CRITIQUE
- Token synchronisé ou double-submit sur toute mutation, en plus de la vérification `Origin` / `Sec-Fetch-Site`.
- ☐ **Aucune mutation en GET** CRITIQUE
- Les GET ne modifient jamais d'état (suppression, déconnexion, paiement...).
- ☐ **Rotation de session** HAUTE
- Nouvel identifiant après login, élévation de privilèges et changement de mot de passe (session fixation).
- ☐ **Expiration** HAUTE
- Timeout d'inactivité + durée absolue. Révocation côté serveur au logout et au reset du mot de passe.
- ☐ **Tokens hors du localStorage** HAUTE
- Les tokens d'auth restent en cookie `HttpOnly` : tout ce qui est en `localStorage` est lisible par une XSS.



07 · A07 · AUTHENTICATION FAILURES

Authentification

___ / 9

- ☐ **Hash des mots de passe** CRITIQUE
- Argon2id (ou bcrypt coût ≥ 12 , scrypt). Jamais MD5, SHA-1 ou SHA-256 seul.
- ☐ **Anti brute-force** CRITIQUE
- Rate limiting par IP et par compte, backoff progressif, CAPTCHA/Turnstile après N échecs.

- ☐ **MFA** CRITIQUE
Obligatoire pour les admins, proposée à tous. Passkeys/WebAuthn ou TOTP plutôt que le SMS.
- ☐ **Pas d'énumération de comptes** HAUTE
Message et temps de réponse identiques pour « compte inconnu » et « mauvais mot de passe », au login, au reset et à l'inscription.
- ☐ **Reset de mot de passe** HAUTE
Token aléatoire ≥ 128 bits, à usage unique, expirant en 15-60 min, stocké hashé. Sessions invalidées après le reset.
- ☐ **Politique de mot de passe** HAUTE
Minimum 12 caractères, refus des mots de passe compromis (API HIBP k-anonymity), pas de règles de composition absurdes.
- ☐ **JWT** HAUTE
Algorithme épinglé côté serveur (refuser `none` et la confusion HS/RS), `exp` court, `aud` et `iss` vérifiés, refresh token rotatif.
- ☐ **OAuth / OIDC** HAUTE
PKCE, paramètre `state` vérifié, `redirect_uri` en correspondance exacte, `nonce` contrôlé.
- ☐ **Comptes par défaut** RECO
Aucun compte admin/admin ni identifiant de démo en production.



08 · A01 · BROKEN ACCESS CONTROL

Contrôle d'accès

__ / 7

Catégorie n°1 de l'OWASP Top 10, et la plus fréquente dans les rapports d'audit.

- ☐ **Autorisation côté serveur à chaque requête** CRITIQUE
Masquer un bouton ne protège rien : chaque endpoint vérifie le rôle ET la propriété de la ressource.
- ☐ **IDOR / BOLA** CRITIQUE
Tester `/api/invoices/123` vers `124` avec un autre compte. Filtrer les requêtes par `owner_id = session.user`.
- ☐ **Deny by default** CRITIQUE
Toute nouvelle route est privée sauf mention contraire, avec un middleware global d'authentification.
- ☐ **Mass assignment** HAUTE
Liste blanche des champs modifiables : interdire `role`, `is_admin` et `price` dans le body.
- ☐ **Escalade horizontale et verticale** HAUTE
Test croisé user A vers user B, user vers admin, tenant A vers tenant B.

- ☐ **Row Level Security** HAUTE
- Si Supabase/Postgres : RLS activée sur toutes les tables exposées, politiques testées.
- ☐ **Back-office isolé** RECO
- Admin sur un sous-domaine ou chemin séparé, restreint par IP/VPN/SSO si possible.

Coffee Beans

09 · A01 · A02 · A05

API, CORS & SSRF

__ / 9

- ☐ **CORS strict** CRITIQUE
- Liste blanche explicite des origines. Jamais `*` avec `credentials`, jamais de reflet aveugle du header `Origin`.
- ☐ **Validation des entrées par schéma** CRITIQUE
- Zod, Joi ou JSON Schema sur chaque endpoint : types, longueurs, formats, champs inconnus rejetés.
- ☐ **SSRF** CRITIQUE
- Les URL fournies par l'utilisateur passent par une liste blanche. Blocage de `169.254.169.254`, `localhost`, des IP privées et des redirections. IMDSv2 sur AWS.
- ☐ **Rate limiting & quotas** HAUTE
- Par IP, utilisateur et clé API, avec une réponse `429` + `Retry-After`.
- ☐ **Taille des payloads** HAUTE
- Limite sur le body, la pagination (`limit` max), les uploads et la profondeur JSON.
- ☐ **Erreurs génériques** HAUTE
- Aucune stack trace, requête SQL ou chemin serveur dans la réponse. Le détail reste dans les logs.
- ☐ **GraphQL** HAUTE
- Introspection coupée en prod, limites de profondeur et de complexité, batching limité.
- ☐ **Webhooks entrants** HAUTE
- Signature HMAC vérifiée, horodatage contrôlé (anti-replay), idempotence.
- ☐ **Inventaire des endpoints** RECO
- OpenAPI à jour, aucun endpoint « shadow » ni ancienne version `/v1` oubliée.

Coffee Beans

10 · A05 · A06

Upload de fichiers

__ / 7

- ☐ **Validation du type réel** CRITIQUE
- Magic bytes vérifiés côté serveur, pas seulement l'extension ou le `Content-Type` envoyé.

- ☐ **Stockage hors webroot** CRITIQUE
Bucket privé ou domaine séparé (ex. `usercontent...`). Jamais exécutable par le serveur.
- ☐ **Renommage** HAUTE
Nom aléatoire (UUID) : on n'utilise jamais le nom fourni pour le chemin.
- ☐ **Taille & quantité** HAUTE
Limites strictes, plus des protections contre les zip bombs et les images géantes.
- ☐ **Servis en téléchargement** HAUTE
`Content-Disposition: attachment` + `nosniff` pour les types non affichables.
- ☐ **Antivirus** RECO
Scan ClamAV ou service cloud avant mise à disposition. Métadonnées EXIF retirées des images.
- ☐ **URL signées** RECO
Accès aux fichiers privés via des URL signées à courte durée de vie.

Secrets & configuration

- ☐ **Aucun secret dans le code** CRITIQUE
gitleaks/trufflehog sur tout l'historique Git + hook pre-commit. Rotation de tout secret ayant fuité.
- ☐ **Fichiers sensibles inaccessibles** CRITIQUE
`/.env`, `/.git/`, `/backup.sql`, `/config.php.bak` et `/.DS_Store` renvoient 404.
- ☐ **Mode debug coupé** CRITIQUE
`NODE_ENV=production`, `DEBUG=False`. Pas de profiler, phpinfo ou console Werkzeug exposés.
- ☐ **Secrets côté client** HAUTE
Seules les clés publiques sont dans le bundle front (`PUBLIC_`, `NEXT_PUBLIC_`, `VITE_`). Pas de `service_role` ni de clé secrète.
- ☐ **Source maps** HAUTE
Non publiées en prod, ou réservées à Sentry.
- ☐ **Listing de répertoires** HAUTE
Désactivé sur le serveur et sur les buckets.
- ☐ **Gestion des secrets** HAUTE
Variables d'environnement de la plateforme ou coffre (AWS Secrets Manager, Doppler, 1Password). Rotation planifiée.

- ☐
- Moindre privilège des clés**
- RECO
- Une clé par service, avec des scopes minimaux et des restrictions IP/domaine (Google Maps, etc.).

Coffee Beans

12 · A03 · SOFTWARE SUPPLY CHAIN FAILURES

Dépendances & supply chain

___ / 8

- ☐
- Vulnérabilités connues**
- CRITIQUE
- `pnpm audit` / `npm audit` à zéro high/critical. Dependabot ou Renovate actif.

- ☐
- Lockfile commité**
- CRITIQUE
- Installation reproductible : `pnpm install --frozen-lockfile` en CI.

- ☐
- SRI sur les scripts CDN**
- HAUTE
- `integrity="sha384-..."` `crossorigin="anonymous"` sur tout script ou CSS tiers.

- ☐
- GitHub Actions épinglées**
- HAUTE
- Actions tierces épinglées par SHA, `permissions:` minimales, aucun secret exposé aux PR de forks.

- ☐
- Scripts d'install**
- HAUTE
- Scripts `postinstall` limités (`onlyBuiltDependencies` dans pnpm). Méfiance envers les paquets récents ou peu suivis.

- ☐
- Images Docker**
- HAUTE
- Image de base minimale (distroless/alpine), utilisateur non-root, scan Trivy/Grype.

- ☐
- SBOM**
- RECO
- Nomenclature logicielle générée (CycloneDX/SPDX) à chaque release.

- ☐
- Composants abandonnés**
- RECO
- Remplacer les libs sans maintenance depuis plus de 2 ans et les runtimes en fin de vie.

Coffee Beans

13 · A02 · SECURITY MISCONFIGURATION

Infrastructure & cloud

___ / 10

- ☐
- Comptes cloud**
- CRITIQUE
- MFA sur root/owner, root jamais utilisé au quotidien, IAM au moindre privilège, pas de clés d'accès longue durée.

- ☐
- Bases de données privées**
- CRITIQUE
- Aucune DB, Redis ou Elasticsearch exposé sur Internet. Security groups restreints, accès via bastion/VPN.

- ☐ **Buckets** CRITIQUE
- S3 Block Public Access activé au niveau du compte. Politiques auditées.
- ☐ **WAF & anti-DDoS** HAUTE
- Cloudflare, AWS WAF ou équivalent, règles OWASP managées, rate limiting en bordure.
- ☐ **Chiffrement au repos** HAUTE
- Disques, DB, backups et buckets chiffrés (KMS).
- ☐ **Patching** HAUTE
- OS et runtimes à jour, mises à jour de sécurité automatiques, plan pour les CVE critiques sous 72 h.
- ☐ **SSH** HAUTE
- Clé uniquement, root login interdit, port fermé au public (SSM ou bastion).
- ☐ **Sauvegardes 3-2-1** HAUTE
- Chiffrées, hors du compte principal, immuables. Restauration testée chaque trimestre.
- ☐ **Ports ouverts** RECO
- Scan externe (nmap, Shodan) : seuls 80 et 443 répondent.
- ☐ **Environnements séparés** RECO
- Prod, staging et dev isolés (comptes, données, secrets). Aucune donnée réelle hors prod.

Logs, détection & réponse

__ / 7

- ☐ **Événements de sécurité journalisés** CRITIQUE
- Logins (succès et échecs), changements de droits, resets, accès admin, erreurs 401/403 en masse.
- ☐ **Pas de données sensibles dans les logs** CRITIQUE
- Mots de passe, tokens, numéros de carte et données de santé masqués.
- ☐ **Centralisation & alertes** HAUTE
- Logs centralisés (CloudWatch, Datadog, Grafana Loki), alertes sur les anomalies, rétention ≥ 1 an pour la sécurité.
- ☐ **Monitoring d'erreurs** HAUTE
- Sentry ou équivalent, avec scrubbing des données personnelles.
- ☐ **Gestion des erreurs** HAUTE
- Les exceptions échouent en mode fermé (fail closed) et ne laissent pas d'état incohérent (A10).

☐ **Plan de réponse à incident** HAUTE
Qui appeler, comment couper un accès, révoquer des clés, communiquer. Exercice annuel.

☐ **Intégrité des logs** RECO
Logs en append-only, non modifiables par l'application.



15 · CNIL · RGPD

RGPD & conformité

__ / 7

☐ **Consentement cookies** CRITIQUE
Aucun traceur non essentiel avant consentement. Bouton « Refuser » aussi visible que « Accepter ».

☐ **Violation de données** CRITIQUE
Procédure de notification à la CNIL sous 72 h + registre des violations.

☐ **Registre des traitements** HAUTE
À jour : finalités, bases légales, durées de conservation, sous-traitants.

☐ **Sous-traitants (DPA)** HAUTE
Contrats signés, hébergement UE ou garanties de transfert (DPF/CCT).

☐ **Droits des personnes** HAUTE
Accès, rectification, suppression et portabilité traités sous 1 mois, avec suppression effective (backups compris).

☐ **Minimisation & purge** HAUTE
On ne collecte que le nécessaire. Purge automatique à l'échéance.

☐ **Mentions légales & politique de confidentialité** RECO
À jour, accessibles depuis toutes les pages et les formulaires.

Config prête à copier

En-têtes de sécurité : `_headers` (Netlify / Cloudflare Pages)

Le nonce CSP doit être généré à chaque requête (middleware / edge function). Pour un site 100 % statique, utilisez des hashes `'sha256-...'` à la place.

```
/*
  Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
  Content-Security-Policy: default-src 'self'; script-src 'self' 'nonce-{NONCE}' 'strict-dynamic'; style
-src 'self' https://fonts.googleapis.com; font-src https://fonts.gstatic.com; img-src 'self' data; conn
ect-src 'self'; object-src 'none'; base-uri 'none'; frame-ancestors 'none'; form-action 'self'; upgrade-
insecure-requests
  X-Content-Type-Options: nosniff
  X-Frame-Options: DENY
  Referrer-Policy: strict-origin-when-cross-origin
  Permissions-Policy: camera=(), microphone=(), geolocation=(), payment=(), usb=()
  Cross-Origin-Opener-Policy: same-origin
  Cross-Origin-Resource-Policy: same-origin
```

Enregistrements DNS de référence

```
example.fr.      CAA  0 issue "letsencrypt.org"
example.fr.      CAA  0 iodef "mailto:security@example.fr"
example.fr.      TXT  "v=spf1 include:_spf.google.com -all"
_dmarc.example.fr. TXT  "v=DMARC1; p=reject; rua=mailto:dmarc@example.fr; adkim=s; aspf=s"
_mta-sts.example.fr. TXT  "v=STSV1; id=20261003"
_smtp._tls       TXT  "v=TLSRPTv1; rua=mailto:tlsrpt@example.fr"
# domaine qui n'envoie aucun mail
parked.fr.       MX   0 .
parked.fr.       TXT  "v=spf1 -all"
```

Vérifications rapides en ligne de commande

```
# En-têtes
curl -sI https://example.fr | grep -iE 'strict|content-security|x-frame|x-content|referrer|permissions'
# TLS
testssl.sh --severity MEDIUM example.fr
# DNS & mail
dig +short CAA example.fr ; dig +short TXT _dmarc.example.fr ; dig +dnssec example.fr
# Fichiers exposés
for p in .env .git/HEAD .DS_Store backup.sql; do curl -so /dev/null -w "%{http_code} $p\n" https://examp
le.fr/$p; done
# Secrets & dépendances
gitLeaks detect --source . ; pnpm audit --audit-level high
```

Boîte à outils de l'auditeur

Les outils que les auditeurs lancent en premier. Passez-les vous-même avant eux.

securityheaders.com

En-têtes HTTP

developer.mozilla.org/observatory

En-têtes + bonnes pratiques

ssllabs.com/ssltest

TLS

testssl.sh

TLS en CLI

internet.nl

DNSSEC, mail, IPv6, TLS

dnsviz.net

DNSSEC

mxttoolbox.com

SPF, DKIM, DMARC, blacklists

hstspreload.org

HSTS preload

csp-evaluator.withgoogle.com

Analyse de CSP

OWASP ZAP / Burp Suite

Scan dynamique (DAST)

nuclei

Scan de templates CVE et misconfig

Semgrep / CodeQL

Analyse statique (SAST)

gitleaks / trufflehog

Secrets dans Git

Trivy / Gype

Images et dépendances

dnstwist

Typosquatting

crt.sh

Certificate Transparency

**Coffee
Beans**

Besoin d'un pré-audit ?

Coffee Beans passe votre application au crible de cette checklist et corrige les points bloquants avant l'audit officiel.

coffee-beans.fr