Protéger un site web contre le scraping, les robots et les agents IA

Menaces, techniques de protection, de contournement et stratégies de défense en profondeur

LinuXMaine.org

Juin 2026

Contents

Avant-propos

Ce document propose un panorama technique des moyens de protection des sites web contre la collecte automatisée de données (*scraping*), les robots d'indexation (*spiders*, *crawlers*) et la nouvelle génération de robots exploités par les modèles d'intelligence artificielle (*AI bots*, agents autonomes).

Il s'adresse aux administrateurs systèmes, aux DevOps, aux développeurs web, aux responsables sécurité (RSSI) et à toute personne désireuse de comprendre les enjeux de la circulation automatisée sur le web.

Le parti pris est résolument **défensif** : les techniques de contournement décrites ne le sont que pour permettre aux défenseurs de comprendre les capacités réelles des attaquants et de calibrer leurs protections en conséquence. Aucune de ces informations ne doit être utilisée pour collecter des données sans autorisation, contourner des mesures de sécurité ou violer les conditions d'utilisation d'un service tiers.

> **Avertissement juridique.** En France et dans l'Union européenne, l'accès et l'extraction non autorisés de données peuvent relever de plusieurs régimes : la loi Godfrain (atteinte aux systèmes de traitement automatisé de données), le droit *sui generis* des bases de données, le droit d'auteur, le RGPD pour les données personnelles, et le droit des contrats (CGU). Voir le chapitre dédié.

Introduction : le web, les humains et les machines

Le web a été conçu pour des êtres humains qui lisent des pages dans un navigateur. Or, une part majoritaire du trafic mondial est aujourd'hui générée par des **programmes automatisés**. Selon les rapports annuels du secteur (Imperva/Thales *Bad Bot Report*, Cloudflare Radar), les robots représentent désormais autour de la moitié du trafic web, dont une fraction importante de robots malveillants.

L'arrivée massive des modèles de langage (LLM) et des agents IA a profondément modifié le paysage en 2023-2026 :

- de nouveaux robots collectent du contenu pour **entraîner** des modèles (GPTBot, ClaudeBot, Google-Extended, CCBot de Common Crawl, etc.) ;
- des agents IA naviguent en **temps réel** pour répondre à des requêtes utilisateur (ChatGPT-User, PerplexityBot, Claude-Web/Claude-User) ;
- des frameworks d'agents autonomes exécutent des navigateurs *headless* pilotés par IA et reproduisent un comportement quasi humain.

Cette évolution pose des questions inédites : consentement à l'entraînement, coûts d'infrastructure, propriété intellectuelle, mais aussi **découvrabilité** (un site bloquant tous les robots IA disparaît des réponses des assistants).

Ce document est organisé en sept grandes parties :

1. une **taxonomie** des robots et agents ;
2. les **problèmes** concrets rencontrés par les sites ;
3. les **techniques de protection**, des plus simples aux plus avancées ;
4. un **focus** sur Anubis et HAProxy pour bloquer les robots IA ;
5. la question des **agents IA et du Markdown** (`llms.txt`, markdown.new) ;
6. les **techniques de contournement** (perspective défensive) ;
7. les aspects **juridiques** et les **bonnes pratiques** stratégiques.

Partie 1 — Taxonomie : robots, spiders, scrapers et agents IA

1.1 Vocabulaire

Terme	Définition
Bot / robot	Programme automatisé émettant des requêtes HTTP sans intervention humaine directe.
Crawler / spider	Robot qui suit les liens pour parcourir et indexer le web (ex. Googlebot).
Scraper	Robot spécialisé dans l'**extraction** structurée de données d'une ou plusieurs pages.
Agent IA	Programme piloté par un modèle de langage, capable de naviguer et d'agir de façon autonome.
Headless browser	Navigateur sans interface graphique (Chromium, Firefox) piloté par script.
Scraping farm	Infrastructure distribuée (proxies, conteneurs) dédiée à la collecte massive.

1.2 Bons robots, mauvais robots

On distingue classiquement :

- **Bons robots déclarés et utiles** : moteurs de recherche (Googlebot, Bingbot, DuckDuckBot), robots de réseaux sociaux (génération de cartes de prévisualisation Open Graph), robots de surveillance (UptimeRobot), archivage (`ia_archiver` d'Internet Archive), robots d'accessibilité.
- **Robots « gris »** : agrégateurs de prix, robots SEO (AhrefsBot, SemrushBot, MJ12bot), robots IA d'entraînement. Légitimes pour certains, indésirables pour d'autres : tout dépend du modèle économique du site.
- **Robots malveillants** : *scrapers* de contenu, robots de *credential stuffing*, scanners de vulnérabilités, robots de spam de formulaires, *scalpers* (achat automatisé de stock), robots de fraude publicitaire.

1.3 Les robots de l'écosystème IA

Robot (User-Agent)	Éditeur	Finalité
`GPTBot`	OpenAI	Entraînement des modèles
`ChatGPT-User`	OpenAI	Navigation à la demande (agent)
`OAI-SearchBot`	OpenAI	Indexation pour la recherche
`ClaudeBot`	Anthropic	Entraînement / indexation
`Claude-User`	Anthropic	Récupération à la demande
`Google-Extended`	Google	Jeton de consentement entraînement (Gemini)
`Googlebot`	Google	Indexation recherche classique
`CCBot`	Common Crawl	Corpus public d'entraînement
`PerplexityBot`	Perplexity	Indexation et réponses
`Bytespider`	ByteDance	Entraînement (réputé agressif)
`Amazonbot`, `Applebot-Extended`	Amazon, Apple	Collecte / entraînement

Note. Les agents « à la demande » (comme `ChatGPT-User`) posent un problème particulier : ils agissent au nom d'un humain, ce qui complique la distinction entre trafic légitime et abus.

Partie 2 — Les problèmes rencontrés par les sites web

2.1 Charge et coûts d'infrastructure

Le scraping massif génère un volume de requêtes qui peut :

- saturer les serveurs applicatifs et la base de données (effet **DDoS involontaire**) ;
- exploser la facture de **bande passante** et de calcul (cloud, CDN) ;
- dégrader l'expérience des utilisateurs réels (latence, indisponibilité).

De nombreux administrateurs de projets open source (forges Git, wikis, GitLab/Gitea) ont signalé en 2024-2025 des pics de trafic provoqués par les robots IA, au point de rendre leurs services instables.

2.2 Vol de contenu et propriété intellectuelle

- **Duplication de contenu** : republication d'articles, ce qui peut nuire au référencement (contenu dupliqué) et à la marque.
- **Entraînement IA non consenti** : utilisation d'œuvres protégées pour entraîner des modèles commerciaux sans rémunération ni attribution.
- **Désintermédiation** : les assistants IA répondent directement à partir du contenu, privant le site des visites et donc des revenus publicitaires.

2.3 Extraction de données métier

- **Scraping de prix** (e-commerce, voyages) par les concurrents.
- **Extraction de bases de données** : annuaires, petites annonces, profils.
- **Collecte de données personnelles** (e-mails, profils) — souvent en violation du RGPD.

2.4 Fraude et abus applicatifs

- **Credential stuffing** : test massif de couples identifiant/mot de passe volés sur les pages de connexion.
- **Account takeover** (ATO), création de **faux comptes**.
- **Card testing** : validation de numéros de cartes volés via des micro-transactions.
- **Spam de formulaires** (contact, commentaires, inscription).
- **Scalping / inventory hoarding** : réservation ou achat automatisé de stock rare (billetterie, éditions limitées).
- **Fraude publicitaire** : clics et impressions frauduleux.

2.5 Pollution des données et de l'analytique

Le trafic robot fausse les indicateurs (taux de conversion, taux de rebond, A/B testing), ce qui conduit à de mauvaises décisions produit et marketing.

2.6 Le dilemme de la découvrabilité

Bloquer *tous* les robots IA protège le contenu mais peut :

- faire disparaître le site des réponses des assistants (perte de visibilité) ;
- pénaliser le référencement si l'on bloque trop largement les moteurs ;
- bloquer aussi les robots de prévisualisation (cartes sur réseaux sociaux).

La protection est donc toujours un **arbitrage** entre sécurité, coûts, visibilité et expérience utilisateur.

Partie 3 — Techniques de protection

On peut classer les défenses en couches, du déclaratif (poli mais non contraignant) au contraignant (techniquement imposé). Une bonne stratégie combine plusieurs couches : c'est la **défense en profondeur**.

3.1 Mesures déclaratives

3.1.1 robots.txt

Le fichier `robots.txt` (Robots Exclusion Protocol, RFC 9309) indique aux robots *coopératifs* ce qu'ils peuvent ou non explorer.

```
User-agent: GPTBot
Disallow: /

User-agent: CCBot
Disallow: /

User-agent: Google-Extended
Disallow: /

User-agent: *
Disallow: /admin/
Allow: /
```

Limites : `robots.txt` n'est qu'une convention. Les robots malveillants l'ignorent ; il sert surtout à gérer les bons robots et constitue un **signal juridique** d'opposition.

3.1.2 Balises meta robots et en-têtes X-Robots-Tag

```
<meta name="robots" content="noindex, nofollow">
```

```
X-Robots-Tag: noai, noimageai
```

Les directives `noai` / `noimageai` (proposées notamment par DeviantArt) visent à signaler le refus d'usage IA, mais leur respect reste volontaire.

3.1.3 llms.txt et ai.txt

Conventions émergentes pour s'adresser spécifiquement aux LLM (voir Partie 5).

3.2 Filtrage par identité technique

3.2.1 Filtrage par User-Agent

Première barrière, simple mais facilement contournable (un *User-Agent* se falsifie). Exemple avec Nginx :

```
if ($http_user_agent ~* "(GPTBot|CCBot|Bytespider|ClaudeBot)") {
    return 403;
}
```

3.2.2 Vérification des bons robots (Reverse DNS)

Pour autoriser un vrai Googlebot, on vérifie que l'IP correspond bien au domaine de Google (reverse DNS puis forward DNS) ou on s'appuie sur les **plages d'IP publiées** par les éditeurs (Google, Bing, OpenAI publient leurs listes).

3.2.3 Filtrage par IP, ASN et géographie

- Listes noires d'IP réputées (services de *threat intelligence*).
- Blocage par **ASN** (numéro de système autonome) : on bloque des plages appartenant à des hébergeurs cloud (AWS, Hetzner, OVH…) d'où provient beaucoup de trafic robot — à manier avec prudence (faux positifs).
- **Géoblocage** : restreindre l'accès à certains pays.

3.3 Limitation du débit (rate limiting)

Imposer un nombre maximum de requêtes par IP / par session / par jeton sur une fenêtre de temps. Exemples : `limit_req` sur Nginx, *stick-tables* sur HAProxy, *token bucket* applicatif, plugins de *rate limiting* sur les API gateways.

```
limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s;
location /api/ {
    limit_req zone=api burst=20 nodelay;
}
```

Limites : les attaquants distribuent leurs requêtes sur des milliers d'IP (proxies résidentiels), ce qui neutralise le *rate limiting* par IP seule.

3.4 Pièges et leurres (honeypots, tarpits)

- **Honeypot de formulaire** : champ caché en CSS ; s'il est rempli, c'est un robot.
- **Liens-pièges** : liens invisibles interdits dans `robots.txt` ; toute visite trahit un robot qui ignore les règles, et déclenche un bannissement.
- **Tarpit / labyrinthe** : on ralentit volontairement le robot ou on le noie dans des pages générées à l'infini. Exemples notables : **Nepenthes**, **Iocaine**, ou l'« **AI Labyrinth** » de Cloudflare, qui piègent les scrapers IA dans des contenus factices.

3.5 Défis interactifs (CAPTCHA et alternatives)

- **CAPTCHA classiques** : reCAPTCHA (Google), hCaptcha, Friendly Captcha.
- **Cloudflare Turnstile** : défi non intrusif, sans images à résoudre.

- **Limites** : les CAPTCHA dégradent l'expérience, posent des problèmes d'accessibilité, et des **services de résolution** (humains ou IA) les cassent à bas coût. Les modèles de vision récents résolvent une partie des CAPTCHA visuels.

3.6 Empreinte (fingerprinting)

Identifier le client par des caractéristiques difficiles à falsifier :

- **TLS fingerprinting (JA3 / JA4)** : signature de la poignée de main TLS ; les bibliothèques HTTP (curl, requests) ont une empreinte différente d'un vrai navigateur.
- **HTTP/2 fingerprinting** : ordre des trames, paramètres.
- **Empreinte navigateur** : Canvas, WebGL, polices, résolution, `navigator`, détection des incohérences (un *User-Agent* Chrome avec une pile TLS Python).
- **Détection d'automatisation** : marqueurs `navigator.webdriver`, propriétés injectées par Selenium/Puppeteer.

C'est l'une des techniques les plus efficaces contre les robots sophistiqués.

3.7 Analyse comportementale

Au-delà d'une requête isolée, on analyse les **schémas** :

- vitesse et régularité « trop parfaite » de navigation ;
- absence de mouvements de souris / d'événements clavier ;
- séquences d'URL non humaines, taux d'erreur, profondeur de navigation ;
- scoring de risque par apprentissage automatique.

3.8 Pare-feu applicatif (WAF) et services de gestion de bots

- **WAF** : ModSecurity (avec l'OWASP Core Rule Set), AWS WAF, Azure WAF.
- **Services managés de *bot management*** : Cloudflare Bot Management, DataDome, HUMAN (anciennement White Ops), Akamai Bot Manager, Imperva (Thales), Kasada, Netacea. Ils combinent fingerprinting, ML, réputation IP et réseaux de signaux à grande échelle.

3.9 Preuve de travail (Proof of Work)

Imposer au client un calcul coûteux avant l'accès, ce qui rend le scraping massif **économiquement non rentable** sans gêner l'utilisateur occasionnel. C'est le principe d'**Anubis** (voir Partie 4).

3.10 Rendu dynamique et obfuscation

- Génération du contenu en **JavaScript** (le contenu n'est pas dans le HTML initial) — efficace contre les scrapers simples, inutile contre les navigateurs *headless*.
- **Obfuscation** des données (noms de classes aléatoires, contenu chiffré rendu côté client, données en images).
- Inconvénients : nuit à l'accessibilité, au SEO et aux performances.

3.11 Authentification et limitation d'accès

- Mettre le contenu sensible **derrière une authentification**.

- **Paywalls**, quotas par compte, clés d'API avec quotas et facturation.
- Signature des requêtes, jetons à courte durée de vie.

3.12 Tableau récapitulatif

Technique	Coût de mise en œuvre	Efficacité robots simples	Efficacité robots avancés	Impact UX/SEO
robots.txt	Très faible	Faible*	Nulle	Nul
Filtrage User-Agent	Faible	Moyenne	Faible	Nul
Rate limiting	Faible	Bonne	Moyenne	Faible
Honeypots / tarpits	Moyen	Bonne	Bonne	Faible
CAPTCHA	Moyen	Bonne	Moyenne	Élevé
Fingerprinting (JA3/JA4)	Moyen/élevé	Bonne	Bonne	Faible
Analyse comportementale	Élevé	Bonne	Bonne	Faible
Proof of work (Anubis)	Moyen	Bonne	Bonne	Moyen
WAF / Bot management	Élevé	Très bonne	Très bonne	Faible
Authentification	Moyen	Très bonne	Très bonne	Élevé

* Faible car seuls les robots coopératifs le respectent.

Partie 4 — Focus : bloquer les robots IA avec Anubis et HAProxy

4.1 Qu'est-ce qu'Anubis ?

Anubis (projet *TecharoHQ/anubis*, licence MIT, écrit majoritairement en Go) est un **pare-feu applicatif anti-IA** léger qui se place en *reverse proxy* devant un site. Sa devise — « *weigh the soul of incoming HTTP requests* » (peser l'âme des requêtes HTTP) — résume son rôle : évaluer chaque requête et soit la laisser passer, soit la soumettre à un **défi de preuve de travail** (*proof of work*).

Anubis se positionne comme une alternative accessible aux solutions propriétaires (type Cloudflare) pour les sites indépendants, les forges open source et les petites communautés submergées par les scrapers IA.

4.2 Principe de fonctionnement

1. Une requête arrive sur Anubis.
2. Anubis **pondère** la requête selon des règles (*bot policy*) : User-Agent, chemins, en-têtes, réputation.
3. Selon le score, Anubis :
   - laisse passer immédiatement (bons robots explicitement autorisés, ex. Internet Archive) ;
   - **affiche une page intermédiaire** (*interstitial*) qui demande au navigateur d'effectuer un calcul de preuve de travail ;
   - ou bloque.
4. Une fois le défi résolu côté navigateur (en JavaScript), un **cookie** signé est délivré et l'utilisateur accède au site sans nouveau défi pendant la durée de validité.

La preuve de travail rend le *crawling* massif **coûteux en CPU** pour l'attaquant, tout en restant imperceptible pour un visiteur humain qui ne charge que quelques pages.

4.3 Avantages et limites

Avantages

- Léger, open source (MIT), auto-hébergeable, sans dépendance à un tiers.
- Politiques de bots configurables (liste d'autorisation des bons robots).
- Efficace contre la majorité des scrapers IA actuels.

Limites

- **Dépend de JavaScript** : exclut les utilisateurs ayant désactivé JS et, par extension, les moteurs de recherche classiques. Ben Tasker recommande donc de ne déployer Anubis

que sur des parties de site qui **nécessitent déjà JavaScript**, afin de ne pas pénaliser le référencement d'une page d'accueil statique.
- Réponse « **nucléaire** » : efficace mais potentiellement brutale (risque de bloquer du trafic légitime).
- La preuve de travail consomme un peu de CPU côté client.

4.4 Déploiement type (d'après le retour d'expérience de Ben Tasker)

Architecture en couches de proxys :

```
Utilisateur
   |
   ▼
Nginx (terminaison TLS)
   |
   ▼
Anubis (preuve de travail)
   |
   ▼
Nginx (routage)  →  Service d'origine (backend)
```

Déploiement avec **Docker Compose** :

```yaml
services:
  anubis:
    image: ghcr.io/techarohq/anubis:latest
    restart: unless-stopped
    environment:
      BIND: ":8923"
      TARGET: "http://127.0.0.1:8023"   # backend protégé
      COOKIE_DOMAIN: "exemple.org"
      OG_PASSTHROUGH: "true"            # laisse passer les aperçus Open Graph
      OG_EXPIRY_TIME: "1h"
      DIFFICULTY: "4"                   # difficulté de la preuve de travail
      SERVE_ROBOTS_TXT: "true"
    network_mode: host                  # avec des règles de pare-feu strictes
```

Points clés relevés :

- image `ghcr.io/techarohq/anubis:latest` ;
- `TARGET` pointe vers le backend (ici `http://127.0.0.1:8023`) ;
- `OG_PASSTHROUGH` préserve les **prévisualisations** sur les réseaux sociaux ;
- `COOKIE_DOMAIN` pilote le suivi de session du cookie de validation ;
- en mode *host networking*, on durcit le **pare-feu** pour n'exposer que ce qui est nécessaire.

4.5 Intégration avec HAProxy

Bien que l'exemple de Ben Tasker utilise Nginx, Anubis s'intègre tout aussi bien derrière **HAProxy**, très répandu pour la répartition de charge. Deux schémas possibles :

Schéma A — HAProxy en frontal, Anubis en backend conditionnel

HAProxy effectue un premier tri (ACL sur User-Agent, géo, *rate limiting* via *stick-tables*) puis route le trafic suspect vers Anubis, et le trafic de confiance directement vers le backend.

```
frontend ft_https
    bind *:443 ssl crt /etc/haproxy/certs/site.pem
    # Détection de robots IA connus
    acl ia_bot hdr_sub(user-agent) -i gptbot ccbot bytespider claudebot
    # Limitation de débit par IP via stick-table
    stick-table type ip size 1m expire 10m store http_req_rate(10s)
    http-request track-sc0 src
    acl trop_de_requetes sc_http_req_rate(0) gt 100

    # Trafic suspect -> Anubis (preuve de travail)
    use_backend bk_anubis if ia_bot or trop_de_requetes
    default_backend bk_app

backend bk_anubis
    server anubis 127.0.0.1:8923

backend bk_app
    server app 127.0.0.1:8023 check
```

Schéma B — Anubis pour tout le trafic HTML

Tout le trafic « pages » passe par Anubis ; les ressources statiques (images, CSS, JS, API authentifiée) sont routées directement, pour ne pas imposer de défi sur chaque sous-ressource.

> **Bonne pratique.** Combiner HAProxy (filtrage rapide, *rate limiting*, terminaison TLS, ACL) **et** Anubis (preuve de travail ciblée) donne une défense graduée : on n'inflige le coût de la preuve de travail qu'au trafic réellement suspect.

4.6 Politiques de bots (bot policies)

Anubis lit des définitions de politiques permettant, par exemple, de **laisser passer explicitement** les bons robots (Internet Archive, moteurs autorisés) tout en imposant le défi aux autres. C'est le bon endroit pour arbitrer entre protection et découvrabilité.

Partie 5 — Agents IA et Markdown : llms.txt et markdown.new

5.1 Le problème : le HTML n'est pas fait pour les machines

« *The web was built for humans. AI agents need structured data.* » Le HTML moderne est lourd (scripts, balises de mise en page, publicités) et coûteux en **jetons** (*tokens*) pour un LLM. Convertir une page en Markdown propre réduit la consommation de jetons d'environ **80 %** : un article de 16 180 jetons en HTML n'en représente plus que 3 150 en Markdown — soit **cinq fois plus de contenu** dans une même fenêtre de contexte.

D'où une approche complémentaire à la protection : plutôt que de seulement *bloquer* les agents IA, on peut **maîtriser et optimiser** ce qu'on leur sert, via deux conventions.

5.2 llms.txt et llms-full.txt

`llms.txt` est un fichier, placé à la racine du site, qui fournit aux LLM une **carte structurée** du site en Markdown : titre, résumé, liens vers les pages importantes et leurs versions Markdown. Objectif : guider l'agent vers le bon contenu, propre et concis, plutôt que de le laisser parser du HTML.

```
# Mon Site

> Documentation et articles sur la sécurité web.

## Articles
- [Protéger son site](https://exemple.org/protection.md): guide complet
- [robots.txt expliqué](https://exemple.org/robots-txt.md): bonnes pratiques

## Optionnel
- [Mentions légales](https://exemple.org/legal.md)
```

On peut aussi publier `llms-full.txt` contenant l'intégralité du contenu en un seul fichier Markdown, et servir une version `.md` de chaque page.

5.3 markdown.new : la conversion à la volée

markdown.new est un service gratuit qui transforme n'importe quelle URL publique en Markdown propre, optimisé pour les agents IA. Il s'utilise de trois façons sans inscription :

- préfixer une URL dans le navigateur : `https://markdown.new/https://...` ;
- envoyer une requête `POST` JSON avec l'URL cible ;
- ajouter des paramètres (`method` = `auto`/`ai`/`browser`, `retain_images` = `true`/`false`).

Pipeline de conversion en trois étages :

1. **Négociation de contenu native** : la requête envoie l'en-tête `Accept: text/markdown` ; un site compatible (ex. derrière Cloudflare) peut renvoyer directement du Markdown depuis l'*edge*, sans parsing.
2. **Repli Workers AI** : si du HTML est renvoyé, la fonction `toMarkdown()` de Cloudflare Workers AI le convertit.
3. **Rendu navigateur** : pour les pages très dynamiques (JS), un navigateur *headless* rend la page avant extraction.

La réponse a le type `text/markdown` et inclut un en-tête `x-markdown-tokens` indiquant le nombre estimé de jetons. Le service gratuit autorise **500 requêtes par jour et par IP** (HTTP 429 au-delà) et peut explorer un site (jusqu'à 100 pages par tâche).

5.4 Reprendre le contrôle : servir soi-même du Markdown

L'enseignement clé pour un éditeur : **mieux vaut servir une version Markdown maîtrisée** que de laisser un tiers parser (et potentiellement déformer) son contenu. Concrètement :

- mettre en place la **négociation de contenu** (`Accept: text/markdown`) et renvoyer une version `.md` propre de chaque article ;
- publier `llms.txt` / `llms-full.txt` ;
- décider en connaissance de cause **qui** peut accéder à ces versions.

5.5 Respecter (ou bloquer) markdown.new

Les éditeurs peuvent **bloquer** markdown.new :

```
User-agent: markdown.new
Disallow: /
```

ou par filtrage du *User-Agent* côté serveur. À l'inverse, ceux qui veulent être bien « lus » par les IA peuvent **faciliter** l'accès Markdown.

> **Arbitrage.** Servir du Markdown améliore la découvrabilité par les assistants IA mais facilite aussi la collecte. Le choix dépend du modèle économique : un site de documentation gagne à être lisible par les IA ; un média sur abonnement cherchera plutôt à restreindre.

Partie 6 — Techniques de contournement (perspective défensive)

> Cette partie décrit, **à des fins de compréhension défensive uniquement**, les méthodes employées par les scrapers sophistiqués. Connaître ces techniques permet de calibrer ses protections. Leur usage pour collecter des données sans autorisation est susceptible d'enfreindre la loi et les CGU.

6.1 Rotation d'adresses IP et proxies

- **Proxies datacenter** : nombreux et bon marché, mais facilement identifiés par ASN.
- **Proxies résidentiels et mobiles** : IP d'utilisateurs réels, très difficiles à distinguer du trafic légitime ; neutralisent le *rate limiting* par IP. C'est le principal défi pour les défenseurs.

Parade : analyse comportementale, fingerprinting, réputation d'IP enrichie, limites par session/compte plutôt que par IP.

6.2 Navigateurs headless et anti-détection

- **Puppeteer**, **Playwright**, **Selenium** pilotent de vrais navigateurs et exécutent le JavaScript ; ils battent le rendu dynamique.
- Outils « furtifs » : `puppeteer-extra-plugin-stealth`, `undetected-chromedriver`, qui masquent `navigator.webdriver` et autres marqueurs d'automatisation.

Parade : empreinte navigateur avancée, détection des incohérences, analyse comportementale (mouvements de souris, *timing*), pièges JS.

6.3 Falsification d'empreinte et d'en-têtes

- *User-Agent* et en-têtes imitant un vrai navigateur.
- **Imitation TLS** : bibliothèques comme `curl-impersonate` reproduisant l'empreinte JA3/JA4 d'un vrai Chrome/Firefox.

Parade : corréler plusieurs signaux (TLS + HTTP/2 + comportement) plutôt qu'un seul ; aucun signal n'est suffisant isolément.

6.4 Résolution de défis

- **Services de résolution de CAPTCHA** (humains à bas coût ou IA) : 2Captcha, Anti-Captcha, etc.

- Résolution de CAPTCHA visuels par modèles de vision.

Parade : préférer les défis sans friction (Turnstile) et la preuve de travail (Anubis), qui visent le **coût** plutôt que la « difficulté » humaine.

6.5 L'escalade permanente

La protection anti-bot est une **course aux armements** : chaque défense suscite un contournement, qui suscite une nouvelle défense. D'où l'importance d'une approche **multi-couches**, évolutive et instrumentée (journaux, métriques, alertes), plutôt que d'une mesure unique.

Partie 7 — Cadre juridique français et européen

Ceci est une synthèse de vulgarisation et ne constitue pas un conseil juridique.

7.1 Atteintes aux systèmes (loi Godfrain)

Les articles 323-1 et suivants du Code pénal répriment l'accès et le maintien frauduleux dans un système de traitement automatisé de données (STAD), ainsi que son entrave. Un scraping qui contourne des mesures de sécurité ou perturbe le service peut tomber sous ce régime.

7.2 Droit des bases de données (droit sui generis)

Le producteur d'une base de données bénéficie d'un droit *sui generis* (articles L.341-1 et suivants du Code de la propriété intellectuelle) lui permettant d'interdire l'**extraction substantielle** de son contenu. Plusieurs décisions (dont l'affaire *leboncoin / Entreparticuliers*) ont sanctionné des extractions massives.

7.3 Droit d'auteur

La reproduction d'œuvres protégées (articles, photos, code) sans autorisation est une contrefaçon. La directive européenne 2019/790 prévoit une exception de **fouille de textes et de données** (*text and data mining*, TDM), assortie d'un droit d'**opposition** (*opt-out*) que les éditeurs peuvent exercer — c'est le fondement juridique des directives `robots.txt`/`noai`.

7.4 Données personnelles (RGPD)

La collecte de données personnelles par scraping doit respecter le RGPD (base légale, information, minimisation). La CNIL a publié des recommandations sur l'aspiration de données et sur l'usage de données pour l'IA. Le scraping d'e-mails ou de profils est souvent illicite.

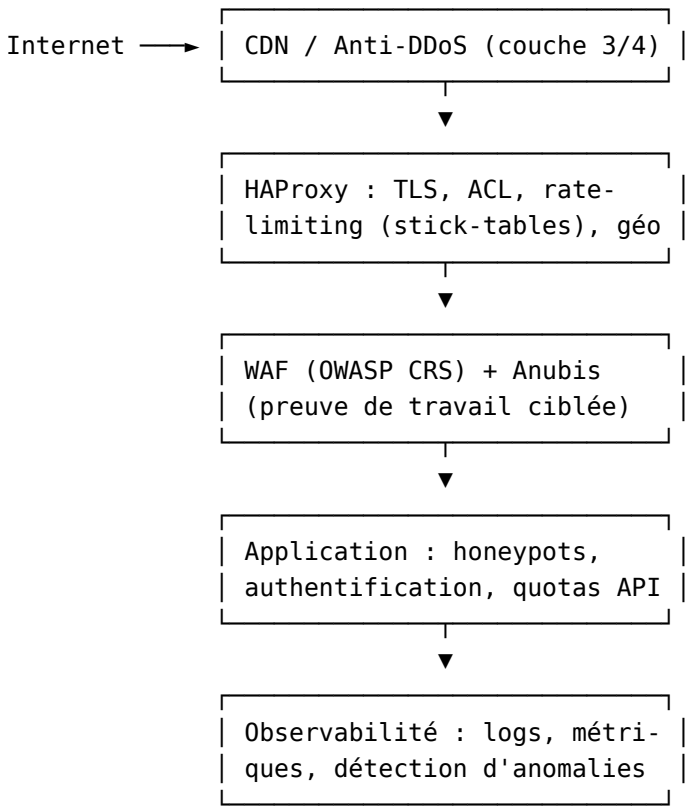
7.5 Droit des contrats (CGU)

Les conditions générales d'utilisation peuvent interdire le scraping. Leur violation engage la responsabilité contractuelle, et l'opposabilité dépend de la manière dont elles ont été acceptées.

Partie 8 — Stratégie : la défense en profondeur

Aucune mesure unique n'est suffisante. Une stratégie robuste empile des couches complémentaires et **mesure** leur efficacité.

8.1 Architecture défensive recommandée



8.2 Checklist opérationnelle

1. Publier un `robots.txt` clair (signal juridique + gestion des bons robots).
2. Déclarer ses intentions IA (`llms.txt`, directives `noai`).
3. Terminer TLS et filtrer tôt (HAProxy : ACL, géo, ASN).
4. Mettre en place un *rate limiting* par session/compte, pas seulement par IP.
5. Déployer un WAF (OWASP CRS) sur les points sensibles.
6. Ajouter une preuve de travail (Anubis) sur les pages HTML coûteuses.
7. Semer des honeypots et surveiller leur déclenchement.

8. Protéger les pages de connexion (MFA, détection de *credential stuffing*).
9. Mettre le contenu de valeur derrière authentification / quotas / API payante.
10. Journaliser, mesurer, alerter, et **itérer** : c'est une course continue.

8.3 Trois profils types

- **Blog / site personnel** : `robots.txt` + Anubis sur les pages HTML + cache CDN. Coût faible, gêne minime.
- **Média / éditeur de contenu** : directives `noai` + bot management + paywall partiel + version Markdown maîtrisée pour la découvrabilité.
- **E-commerce / SaaS** : bot management managé, protection des connexions (anti-ATO), *rate limiting* fin par compte, surveillance de la fraude.

Conclusion

La protection d'un site web contre le scraping, les spiders et les agents IA n'est pas un interrupteur, mais un **équilibre dynamique** entre sécurité, coûts, expérience utilisateur, référencement et découvrabilité par les assistants IA.

L'irruption des LLM a déplacé la frontière : il ne s'agit plus seulement de bloquer des robots, mais aussi de **décider comment l'on souhaite être lu** par les machines — d'où l'intérêt conjoint d'outils défensifs comme **Anubis** (preuve de travail), de filtres robustes comme **HAProxy**, et de conventions d'ouverture maîtrisée comme **llms.txt** et le **Markdown** servi intentionnellement.

La bonne posture combine **défense en profondeur**, **mesure continue** et **arbitrages explicites** alignés sur le modèle économique du site.

Bibliographie technique (en français)

- **CNIL** — *Recommandations sur le développement des systèmes d'intelligence artificielle* et fiches pratiques sur le *web scraping* et les données personnelles. https://www.cnil.fr
- **ANSSI** — *Recommandations pour la sécurisation des sites web* et guides d'hygiène informatique. https://cyber.gouv.fr
- **OWASP** — *OWASP Automated Threats to Web Applications* (projet OAT, taxonomie OAT-001 à OAT-021), versions et ressources traduites. https://owasp.org/www-project-automated-threats-to-web-applications/
- **OWASP** — *Core Rule Set (CRS)* pour ModSecurity, documentation. https://coreruleset.org
- **MDN Web Docs (en français)** — articles sur HTTP, les en-têtes, la négociation de contenu et la sécurité web. https://developer.mozilla.org/fr/
- **HAProxy** — documentation de configuration (ACL, *stick-tables*, *rate limiting*). https://docs.haproxy.org
- **Cloudflare (centre d'apprentissage, FR)** — articles « Qu'est-ce qu'un bot ? », « Gestion des bots », « AI Labyrinth ». https://www.cloudflare.com/fr-fr/learning/bots/
- **Documentation officielle d'Anubis** — TecharoHQ. https://anubis.techaro.lol et https://github.com/TecharoHQ/anubis
- **Spécification du Robots Exclusion Protocol** — RFC 9309. https://www.rfc-editor.org/rfc/rfc9309
- **Proposition llms.txt** — Jeremy Howard. https://llmstxt.org
- **CERT-FR** — bulletins et avis de sécurité. https://www.cert.ssi.gouv.fr

Sélection de livres (en français)

- **Sécurité informatique et réseaux** — Solange Ghernaouti, Dunod. Ouvrage de référence sur les fondamentaux de la cybersécurité.
- **Sécurité web — De l'art de la défense à la pratique de l'attaque**, collection EPSILON / ENI. Panorama des attaques web et des contre-mesures.
- **Sécurité des applications web** (éditions ENI) — vulnérabilités, WAF, durcissement.
- **Hacking et Forensic — Développez vos propres outils en Python** — Franck Ebel, ENI. Comprendre l'outillage offensif pour mieux se défendre.
- **Les bases du hacking** — Patrick Engebretson (trad. fr.), Pearson. Initiation aux tests d'intrusion.
- **Tableaux de bord de la sécurité réseau** — Cédric Llorens, Laurent Levier, Denis Valois, Eyrolles. Supervision et métriques de sécurité.
- **Cybersécurité — Analyser les risques, mettre en œuvre les solutions** — Solange Ghernaouti, Dunod.
- **Architectures de sécurité pour Internet** — éditions Eyrolles, sur la conception d'architectures réseau défensives.
- **Le guide pratique du RGPD** (éditions ENI / Gualino) — pour le volet données personnelles du scraping.

  Les titres et éditions évoluant rapidement, vérifier la dernière édition disponible chez l'éditeur (Dunod, ENI, Eyrolles, Pearson).

Liens pertinents

Outils et projets

- Anubis (dépôt) — https://github.com/TecharoHQ/anubis
- Anubis (documentation) — https://anubis.techaro.lol
- Article de Ben Tasker « Deploying Anubis to block AI bots » — https://www.bentasker.co.uk/posts/blog/the-internet/deploying-anubis-to-block-ai-bots.html
- markdown.new — https://markdown.new/
- llms.txt — https://llmstxt.org
- Nepenthes (tarpit anti-IA) — https://zadzmo.org/code/nepenthes/
- HAProxy — https://www.haproxy.org
- Nginx — https://nginx.org
- ModSecurity / OWASP CRS — https://coreruleset.org
- CrowdSec (détection comportementale open source) — https://www.crowdsec.net
- Fail2ban — https://github.com/fail2ban/fail2ban

Robots IA : documentation des éditeurs

- OpenAI — robots (GPTBot) — https://platform.openai.com/docs/bots
- Google — Google-Extended et robots — https://developers.google.com/search/docs/crawling-indexing/overview-google-crawlers
- Anthropic — ClaudeBot / Claude-User — https://support.anthropic.com
- Common Crawl (CCBot) — https://commoncrawl.org/ccbot
- Perplexity — https://docs.perplexity.ai

Référentiels et standards

- RFC 9309 — Robots Exclusion Protocol — https://www.rfc-editor.org/rfc/rfc9309
- OWASP Automated Threats — https://owasp.org/www-project-automated-threats-to-web-applications/
- OWASP Top 10 — https://owasp.org/www-project-top-ten/
- Spécification JA3/JA4 (fingerprinting TLS) — https://github.com/FoxIO-LLC/ja4

Acteurs du bot management

- Cloudflare Bot Management — https://www.cloudflare.com/fr-fr/application-services/products/bot-management/
- Cloudflare AI Labyrinth — https://blog.cloudflare.com/ai-labyrinth/
- DataDome — https://datadome.co/fr/
- HUMAN Security — https://www.humansecurity.com
- Imperva (Thales) — https://www.imperva.com

Veille et rapports

- Imperva / Thales — *Bad Bot Report* (annuel) — https://www.imperva.com
- Cloudflare Radar — https://radar.cloudflare.com
- CNIL — *web scraping* et IA — https://www.cnil.fr
- ANSSI / cyber.gouv.fr — https://cyber.gouv.fr

---

*Document rédigé par **LinuXMaine.org** — Juin 2026. Diffusé à des fins d'information et de sensibilisation. Les marques citées appartiennent à leurs détenteurs respectifs.*