

Rapport Hackaton :

Rami Taieb	Mamadou Sall
Clement Gibert	Michelle Alvirra Nyadja Kamwa
Achraf Maaloul	Mohamed Amine Dhaoui
Aly Gano	Mohammed Sabar
Amayas Matmar	Navila Spachelle Migue Kamgang
Boris Junior Mahop Mahop	Sarielte Ange Duchelle Yamba
Elyes Boudabous	
Firas Belhiba	
Franck Vaneck Ndandji Tchetnga	

- Avec la montée des examens à distance ou sur ordinateur, les établissements éducatifs font face à un nouveau défi majeur : la triche numérique.
- Les étudiants peuvent facilement accéder à des navigateurs web, des IA comme ChatGPT, des applications de messagerie ou des outils de développement pendant un examen.

 Coûteuses

 Intrusives

 Complexes à déployer à grande échelle

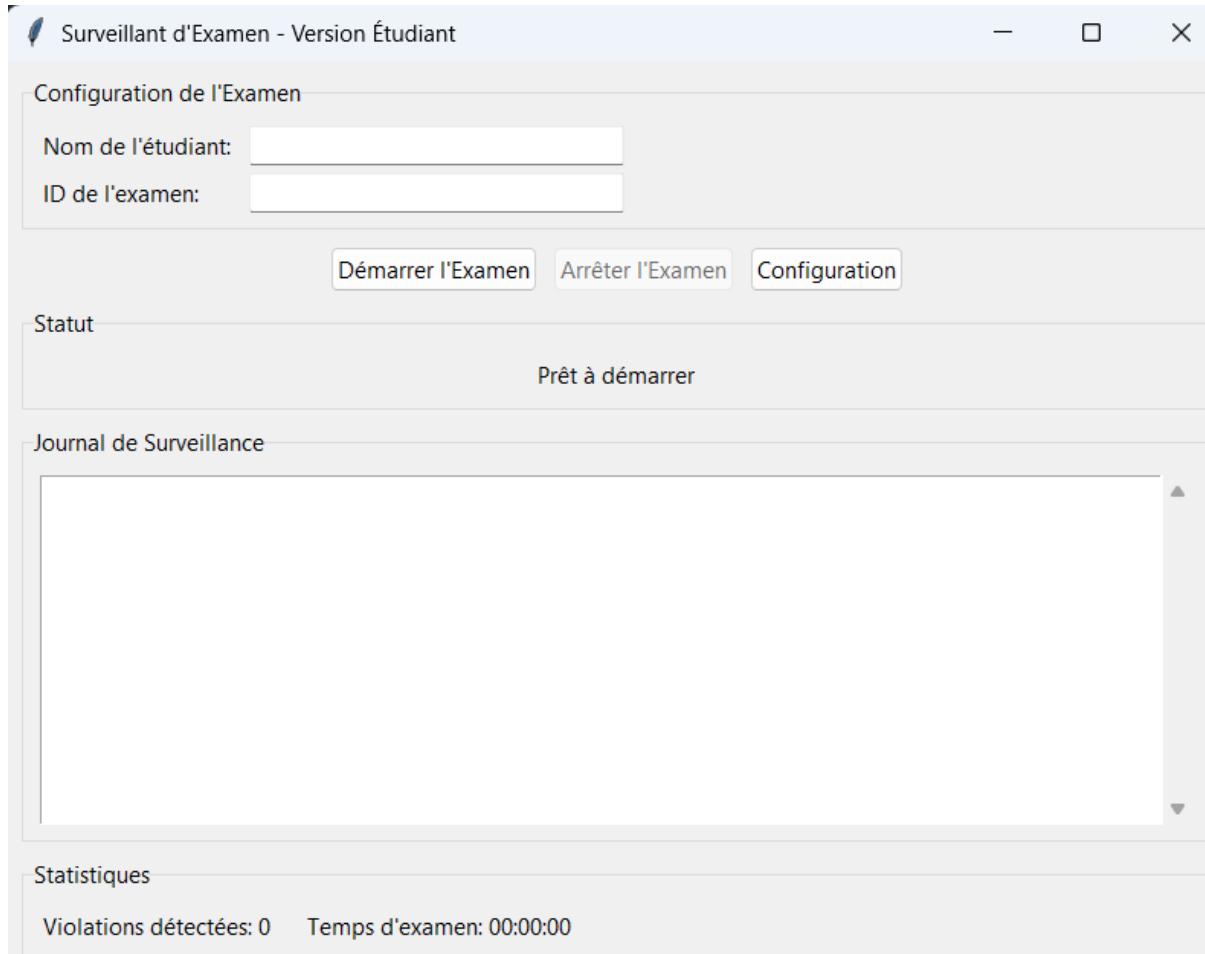
Il devient donc essentiel de concevoir un système de contrôle automatisé pour garantir l'équité, la sécurité, et la fiabilité des épreuves numériques.

- Développer un logiciel anti-triche intelligent, léger et automatisé, capable de :
 - Se déclencher automatiquement à l'heure de l'examen définie par un administrateur.
 - Surveiller en temps réel les activités du PC de l'étudiant
 - Déetecter et signaler tout comportement suspect ou interdit

Classer automatiquement chaque utilisateur en fonction de son comportement :

- normal | suspect | tricheur

Le tout en respectant les bonnes pratiques de confidentialité et en assurant une expérience utilisateur fluide pour l'étudiant comme pour l'administrateur.



L'utilisateur va entrer son nom ainsi que l'ID de l'examen, l'outil va regarder en permanence les applications de triches ainsi que les onglets de chaque navigateur pour voir si l'étudiant triche.

Si l'étudiant triche, un message apparaitra :



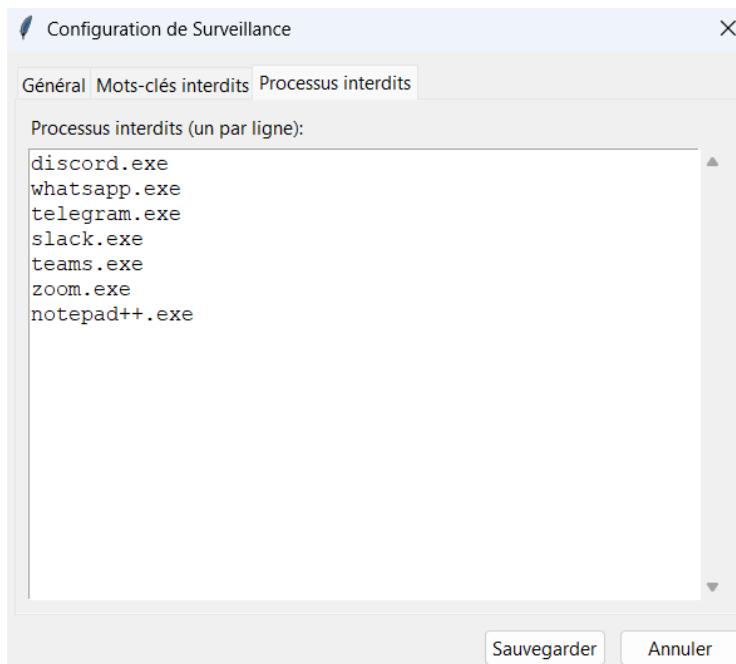
Ainsi que les logs détaillant la triche dans le journal

Journal de Surveillance

```
[12:29:30] INFO: Examen démarré pour monsieur dupon (ID: 01)
[12:29:31] VIOLATION: SITE_INTERDIT: Site interdit détecté: claude dans 'claude - google chrome'
[12:30:01] VIOLATION: SITE_INTERDIT: Site interdit détecté: claude dans 'claude - google chrome'
```

Une fois l'examen terminer, cela créer un rapport sous json mais ce n'est pas tout.

On peut ajouter des mot clé ou des processus ainsi qu'en supprimer depuis l'onglet configuration :



Cela permet de gérer les logiciels que nous voulons détecter comme triche.

L'admin peut ensuite se connecté à un site externe pour avoir le suivi des tricheries qui auront eu lieu pour chaque exam.



Pour l'exam 01 par exemple on trouvera le suivant :

Étudiants ayant participé à l'examen 01

Résultats et violations détectées

7

ÉTUDIANTS

21

VIOLATIONS TOTALES

0

SANS VIOLATION

Nous montrons les statistiques globales :

Clement



2 VIOLATION(S)

clement



4 VIOLATION(S)

Emilien



2 VIOLATION(S)

fr



4 VIOLATION(S)

Ainsi que le nom des étudiants et leurs violations.



Violations détectées

2025-07-02T14:30:02.868735

Processus interdit détecté: code.exe

2025-07-02T14:30:03.810185

Site interdit détecté: google dans 'inscription - google chrome'

← RETOUR À LA LISTE

On a un aperçu détaillé du rapport pour chaque étudiant ce qui permet d'étudier les violations.

Rapport de sécurité pour le projet « Anti-triche IA »

Contexte du projet

Dans le cadre d'un hackathon, nous développons une application de surveillance anti-triche pour les examens

Ce projet vise à :

Déetecter automatiquement la triche lors d'examens en ligne, en identifiant les copier-coller, les changements de fenêtres du navigateur et l'utilisation suspecte d'outils IA.

Il est développé en **Python (FastAPI + JS pour le frontend)** et nécessite une infrastructure sécurisée pour éviter les compromissions qui pourraient invalider la fiabilité du système.

Le rôle de la partie CYBER est de sécuriser l'infrastructure et le code source pour limiter les risques d'exploitation.

Environnement et outils utilisés

- Python 3.11
- Trivy (scan vulnérabilités du code et des images Docker)
- Bandit (analyse sécurité Python)
- auditd (surveillance système)
- UFW, Fail2ban, SSH durci, sysctl
- Headers HTTP, HTTPS, rate limiting
- pip-audit (audit des dépendances Python)
- + recommandations spécifiques pour ton application anti-triche (logs, hash, surveillance JS)

Sécurité du code source

Trivy

Trivy est utilisé pour :

Scanner le code source et les dépendances via :

trivy fs .

Scanner les images Docker (si utilisé) :

docker build -t anti-triche:latest .

trivy image anti-triche:latest

Permet de détecter les vulnérabilités CVE connues dans les bibliothèques système et Python.

Bandit

Analyse avec Bandit sur le projet AntiTriche

Une synthèse des vulnérabilités identifiées par l'outil Bandit lors de l'analyse statique du projet AntiTriche. Il est destiné aux équipes de développement.

1. Résumé Global

- ✓ Lignes de code analysées : 447 853
- ✓ Lignes ignorées via #nosec : 2
- ✓ Nombre total de vulnérabilités détectées : Environ 50+
- ✓ Principales catégories de vulnérabilités identifiées :
 - Usage massif de la fonction assert
 - Utilisation non sécurisée du module subprocess
 - Présence de blocs try/except/pass masquant les erreurs
 - Application Flask lancée avec debug=True en environnement production

2. Vulnérabilité Critique

Flask exécuté avec debug=True

- ✓ Fichier concerné : app.py
- ✓ Ligne : 134
- ✓ Niveau de gravité : Élevé
- ✓ Description : La configuration debug=True active le debugger Werkzeug, ce qui permet l'exécution de code arbitraire par un attaquant.

- ✓ Impact : Risque élevé de compromission complète du serveur en production.
- ✓ Recommandation immédiate : Désactiver impérativement debug=True dans les environnements de production.

3. Vulnérabilités Fréquentes

- ❖ Usage excessif de assert
 - Localisation : Présent dans plusieurs fichiers, notamment dans des modules utilisant PIL.
 - Risque : En mode bytecode optimisé (O), les assertions sont ignorées, ce qui supprime les contrôles critiques.
 - Recommandation : Remplacer les assertions par des vérifications explicites et lever des exceptions contrôlées.
- ❖ Utilisation du module subprocess
 - Fichiers concernés : [AntiTriche.py](#) et plusieurs modules liés à PIL.
 - Risque : Risque d'exécution de commandes externes non sécurisées, potentiellement vulnérable à des injections de commandes.
 - Recommandation : Valider rigoureusement toutes les entrées utilisées dans les appels subprocess et privilégier les appels sécurisés avec chemins absolus et listes d'arguments.
- ❖ Usage de try/except/pass
 - Problème : Présence de blocs try/except avec bloc pass vide, masquant les erreurs et rendant le débogage difficile.
 - Recommandation : Gérer explicitement les exceptions, les logger ou les propager pour garantir la visibilité des erreurs.

4. Plan d’Action Recommendé

- ✓ Court terme
 - Supprimer immédiatement debug=True dans Flask en production.
 - Auditer tous les appels à subprocess et sécuriser les entrées correspondantes.
- ✓ Moyen terme
 - Refactorer le code pour remplacer tous les assert par des contrôles explicites.
 - Éliminer ou corriger les blocs try/except/pass inutiles.
- ✓ Long terme
 - Intégrer l’analyse Bandit dans un pipeline CI/CD pour un contrôle continu.
 - Documenter les bonnes pratiques de sécurité et former les équipes de développement.

Metrics:
Total lines of code: 447853
Total lines skipped (#nosec): 2

Google Translate

```
blacklist: Consider possible security implications associated with the subprocess module.  
Test ID: B404  
Severity: LOW  
Confidence: HIGH  
CWE: CWE-78  
File: /tmp/test.py  
Line number: 26  
More info: https://bandit.readthedocs.io/en/1.8.5/blacklists/blacklist\_imports.html#b404-import-subprocess  
25  
26     import subprocess  
27     import winreg  
  
try: except: pass: Try, Except, Pass detected.  
Test ID: B110  
Severity: LOW  
Confidence: HIGH  
CWE: CWE-703  
File: /tmp/test.py  
Line number: 303  
More info: https://bandit.readthedocs.io/en/1.8.5/plugins/b110\_try\_except\_pass.html  
300         self.last_clipboard = current_clipboard  
301     except:  
302         pass  
303     finally:  
  
flask.debug_true: A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.  
Test ID: B201  
Severity: HIGH  
Confidence: MEDIUM  
CWE: CWE-94  
File: /tmp/app.py  
Line number: 134  
More info: https://bandit.readthedocs.io/en/1.8.5/plugins/b201\_flask\_debug\_true.html  
133     if __name__ == "__main__":  
134         app.run(debug=True)  
  
assert used: Use of assert detected. The enclosed code will be removed when compiling to optimised byte code.  
Test ID: B101  
Severity: LOW  
Confidence: HIGH  
CWE: CWE-703  
File: /venv/lib/python3.12/site-packages/PIL/BipImagePlugin.py  
Line number: 315
```

pip-audit

Audit des dépendances Python (déetecte les CVE dans requirements.txt ou pyproject.toml) :

```
pip install pip-audit  
pip-audit
```

Sécurité du serveur et du système

UFW (pare-feu)

```
sudo ufw default deny incoming  
sudo ufw default allow outgoing  
sudo ufw allow 2222/tcp # SSH sur port custom
```

```
sudo ufw allow 80,443/tcp # HTTP/HTTPS  
sudo ufw enable
```

SSH sécurisé

```
sudo nano /etc/ssh/sshd_config
```

Change le port :

Port 2222

Désactive login root :

PermitRootLogin no

Désactive mot de passe :

PasswordAuthentication no

Puis :

```
sudo systemctl restart ssh
```

sysctl - durcissement noyau

```
sudo tee -a /etc/sysctl.conf <<EOF  
net.ipv4.tcp_syncookies = 1  
net.ipv4.conf.all.rp_filter = 1  
net.ipv4.conf.default.rp_filter = 1  
net.ipv4.icmp_echo_ignore_broadcasts = 1  
EOF  
sudo sysctl -p
```

Fail2ban

Bloque IP après tentatives SSH échouées répétées :

```
sudo apt install fail2ban  
sudo systemctl enable fail2ban  
sudo systemctl start fail2ban
```

Status :

```
sudo fail2ban-client status sshd
```

auditd

Surveillance des fichiers critiques et exécution Python :

```
sudo apt install auditd  
sudo auditctl -w /etc/ssh/sshd_config -p war -k ssh_config
```

```
sudo auditctl -a always,exit -F path=/usr/bin/python3 -F perm=x -k python_exec
```

Pour consulter :

```
sudo ausearch -k ssh_config  
sudo ausearch -k python_exec
```

🌐 Sécurité web (Nginx, HTTPS, Headers)

🔒 Headers HTTP

Dans /etc/nginx/sites-available/default :

```
add_header X-Frame-Options "DENY";  
add_header X-Content-Type-Options "nosniff";  
add_header X-XSS-Protection "1; mode=block";  
add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload";  
add_header Content-Security-Policy "default-src 'self'";  
add_header Referrer-Policy "no-referrer-when-downgrade";
```

🔑 HTTPS avec Certbot

```
sudo apt install certbot python3-certbot-nginx  
sudo certbot --nginx -d monsite.com  
sudo certbot renew --dry-run
```

⌚ Rate limiting

Ajoute dans server {} Nginx :

```
limit_req_zone $binary_remote_addr zone=mylimit:10m rate=5r/s;  
location / {  
    limit_req zone=mylimit burst=10;  
    proxy_pass http://127.0.0.1:8000;  
}
```

📋 Anti-triche spécifique à ton projet Python

🎭 Frontend JS

Déetecte perte de focus (tab hors navigateur) :

```
window.addEventListener("blur", () => {  
    fetch("/api/log", {method: "POST", body: JSON.stringify({event: "blur", timestamp: Date.now()}))}  
});
```

Déetecte copier/coller :

```
document.addEventListener("copy", () => {...});  
document.addEventListener("paste", () => {...});
```

Hash des logs

Chaque session est loguée avec un hash SHA256 pour garantir intégrité :

```
import hashlib  
hash = hashlib.sha256(log_content.encode()).hexdigest()
```

Journalisation structurée

Tout est écrit dans le fichier de journalisation (/var/log/anti-triche/) avec :

- IP (liste des addresses ip)
- Agent navigateur (liste des navigateurs)
- Timestamp (le temps passée)
- Actions suspectes (les activités suspectes)

Toutes les commandes ci-dessus sont regroupées dans un setup.sh qui :

- Met à jour ton OS
- Installe et configure tous ces outils
- Lance un premier audit Bandit et Trivy
- Configure Nginx avec HTTPS + headers

Recommandations complémentaires

Mesure	Pourquoi ?
MFA sur GitHub / GitLab	Sécurise le code source
Tests automatisés sécurité CI	Exécute bandit, trivy à chaque push
Backup chiffré	Garantit restauration propre

Résumé final

Domaine	Mesures
 Réseau	UFW, Fail2ban, SSH sur port custom
 Système	auditd, sysctl hardening
 Code Python	Bandit, pip-audit, Trivy
 Web	Headers HTTP, HTTPS, Rate limit
 Anti-triche	JS logs focus/copy, hash logs

Dans le cadre du durcissement de la plateforme web *AntiTriche*, l'une des premières actions de sécurisation a consisté à modifier le port SSH par défaut utilisé pour l'administration distante du serveur Ubuntu hébergeant l'application.

Édition du fichier de configuration SSH :

J'ai modifié le fichier /etc/ssh/sshd_config

En décommentant et modifiant la ligne suivante :

- #Port 22

+ Port 2520

```
# Site-wide defaults for some commonly used options. For a comprehensive
# list of available options, their meanings and defaults, please see the
# ssh_config(5) man page.

Include /etc/ssh/ssh_config.d/*.conf

Host *
# ForwardAgent no
# ForwardX11 no
# ForwardX11Trusted yes
# PasswordAuthentication yes
# HostbasedAuthentication no
# GSSAPIAuthentication no
# GSSAPIDelegateCredentials no
# GSSAPIKeyExchange no
# GSSAPITrustDNS no
# BatchMode no
# CheckHostIP yes
# AddressFamily any
# ConnectTimeout 0
# StrictHostKeyChecking ask
# IdentityFile ~/.ssh/id_rsa
# IdentityFile ~/.ssh/id_dsa
# IdentityFile ~/.ssh/id_ecdsa
# IdentityFile ~/.ssh/id_ed25519
#
#port 20
Port 2520
```

```
root@ganoaly:~# ufw allow 2520/tcp
```

```
root@ganoaly:~# ufw enable
Firewall is active and enabled on system startup
```

```
root@ganoaly:~# sudo ufw status verbose
^C
root@ganoaly:~# ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), deny (routed)
New profiles: skip

To                      Action      From
--                      -----      ---
2520/tcp                ALLOW IN   Anywhere
2520/tcp (v6)            ALLOW IN   Anywhere (v6)

root@ganoaly:~#
```

Résultat :

Le port SSH par défaut (22) est désormais **fermé**.

Seul le **port 2520** est ouvert pour les connexions SSH.

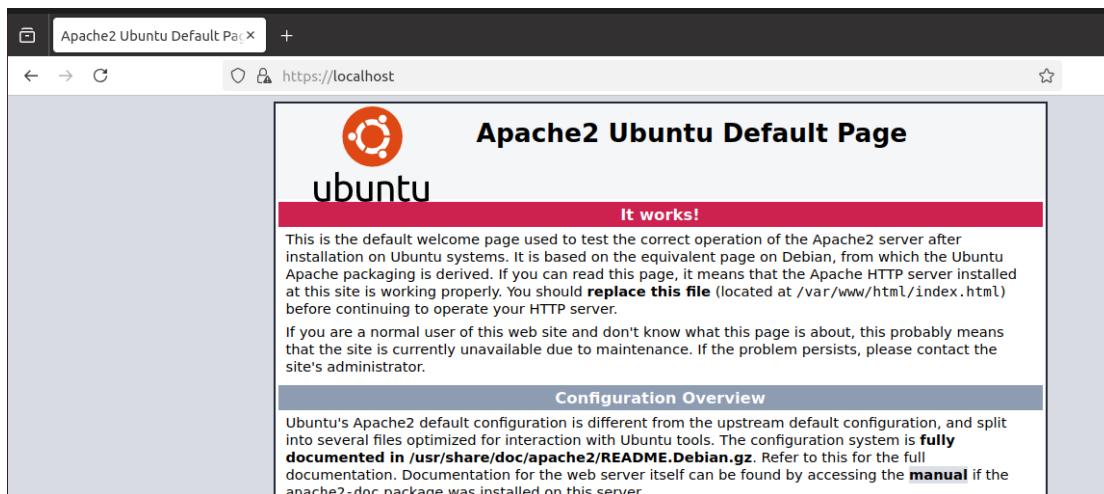
Cette mesure permet de limiter la surface d'attaque du serveur, en empêchant les scripts automatisés de détecter facilement le service SSH.

Mise en place de HTTPS pour sécuriser la plateforme web :

```

root@ganoaly:/etc/apache2/sites-available# a2ensite default-ssl.conf
Enabling site default-ssl.
To activate the new configuration, you need to run:
    systemctl reload apache2
root@ganoaly:/etc/apache2/sites-available#
root@ganoaly:/etc/apache2/sites-available#
root@ganoaly:/etc/apache2/sites-available#
root@ganoaly:/etc/apache2/sites-available# systemctl restart apache2
root@ganoaly:~# a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled
root@ganoaly:~# 
root@ganoaly:~# openssl pkcs12 -export -out certificat.pfx -inkey cle-privee.key -in certificat.crt
Enter Export Password:
Verifying - Enter Export Password:
root@ganoaly:~# 
root@ganoaly:~# openssl req -new -x509 -key cle-privee.key -out certificat.crt -days 365
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:senegal
string is too long, it needs to be no more than 2 bytes long
Country Name (2 letter code) [AU]:sn
State or Province Name (full name) [Some-State]:dk
Locality Name (eg, city) []:rf
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
root@ganoaly:~#
root@ganoaly:~#
root@ganoaly:~# cat cle-privee.key certificat.crt > certificat.pem

```



Activation du service Flask

```
root@ganoaly:/var/www/Hackaton_2025# flask --app app run
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [03/Jul/2025 00:47:55] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [03/Jul/2025 00:47:55] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [03/Jul/2025 01:18:47] "GET /Login HTTP/1.1" 200 -
127.0.0.1 - - [03/Jul/2025 01:18:50] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [03/Jul/2025 01:18:50] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [03/Jul/2025 01:19:02] "POST /login HTTP/1.1" 302 -
127.0.0.1 - - [03/Jul/2025 01:19:02] "GET /dashboard HTTP/1.1" 200 -
127.0.0.1 - - [03/Jul/2025 01:31:28] "POST /dashboard HTTP/1.1" 302 -
```

Inscription administrateur

Nom d'utilisateur :

Mot de passe :

Code Admin : Code secret admin

[S'inscrire](#)

Bienvenue admin (admin)

ID d'examen: [Rechercher](#)

[Déconnexion](#)

Conclusion

Le projet anti-triche Python est désormais sécurisé, audité et prêt pour un déploiement en prod.