RAPPORT TECHNIQUE : LIVRABLE 4

1. Présentation du Scénario (S1) : Empoisonnement de données

Le scénario **S1** simule une attaque par **empoisonnement (Data Poisoning)** ciblant les flux de données provenant des capteurs **OMEGA OS-MINI** (température) des robots **KUKA KR 210**.

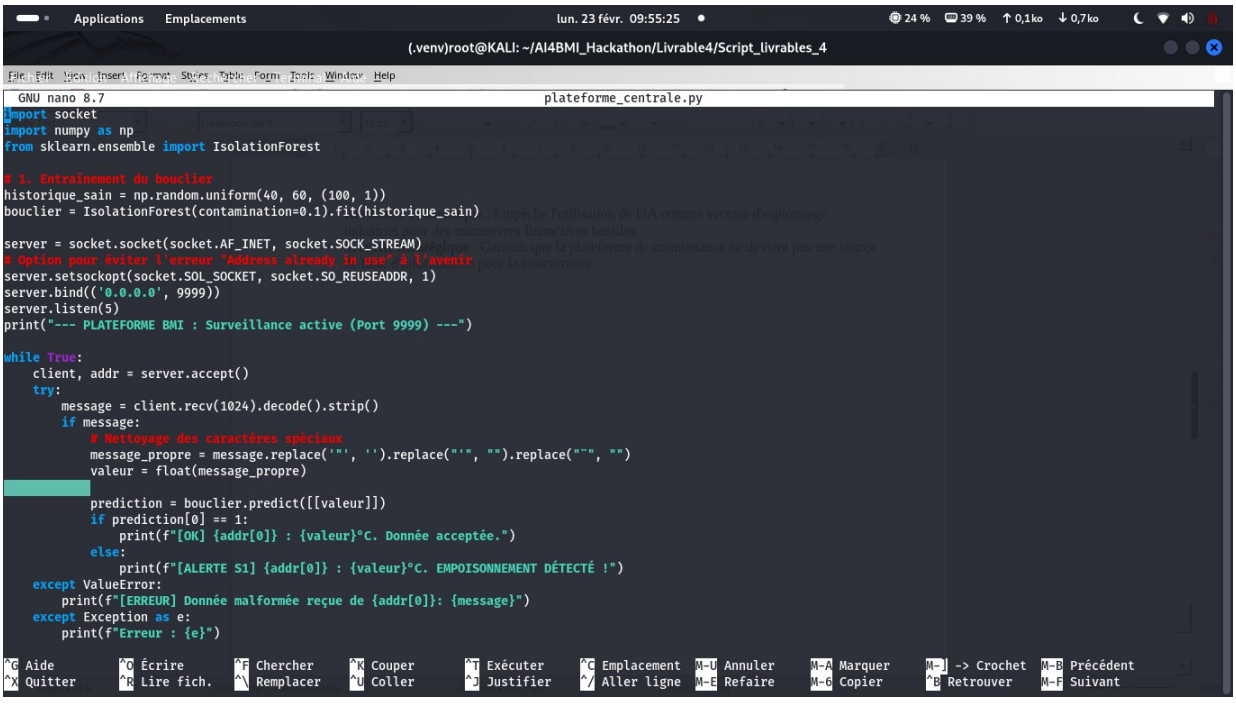
Objectif de l'attaquant : Injecter des relevés de température falsifiés dans le pipeline d'apprentissage pour que l'IA considère des états de surchauffe critique comme étant "normaux". À terme, cela rend le système de maintenance prédictive aveugle aux pannes réelles.

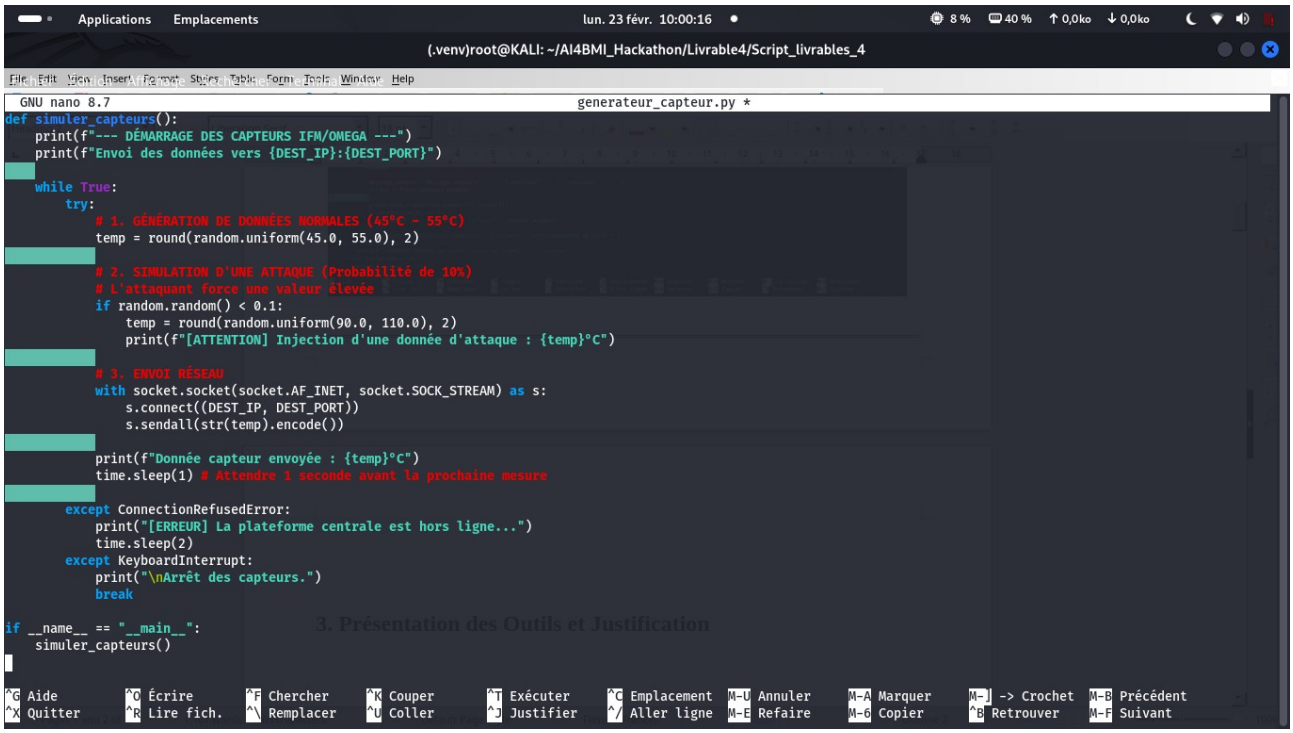
---

2. Architecture de la Simulation

Pour démontrer l'efficacité de notre solution sans matériel physique, nous avons mis en place une architecture réseau virtualisée sur **Kali Linux** :

- **Fichier `plateforme_centrale.py`** : Serveur de défense simulant le centre de données de l'usine BMI. Il intègre le bouclier de sécurité IA.
- **Fichier `generateur_capteurs.py`** : Script simulant un capteur industriel distant qui transmet ses données via le protocole TCP/IP sur le port 9999.
- **Vecteur d'attaque** : Une fonction d'injection malveillante intégrée au simulateur (ou via Netcat) pour envoyer des données hors-normes.





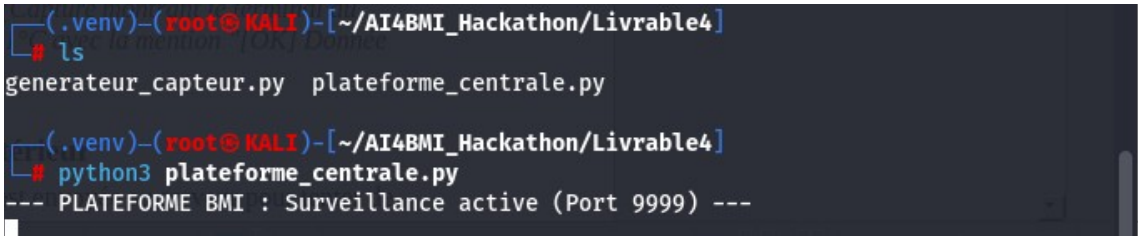
3. Présentation des Outils et Justification

Outil	Rôle dans le projet	Justification
Python 3 / Scikit-Learn	Développement de l'IA	Standard industriel pour le traitement de données massives.
Isolation Forest	Algorithme de défense	Capable de détecter des anomalies sans connaître l'attaque à l'avance (non-supervisé).
Sockets TCP/IP	Communication réseau	Simule fidèlement la couche de transport des données IIoT (Industrie 4.0).
Kali Linux	Environnement de test	Plateforme de référence pour tester la résilience des systèmes.

4. Démonstration Technique (Étapes et Captures)

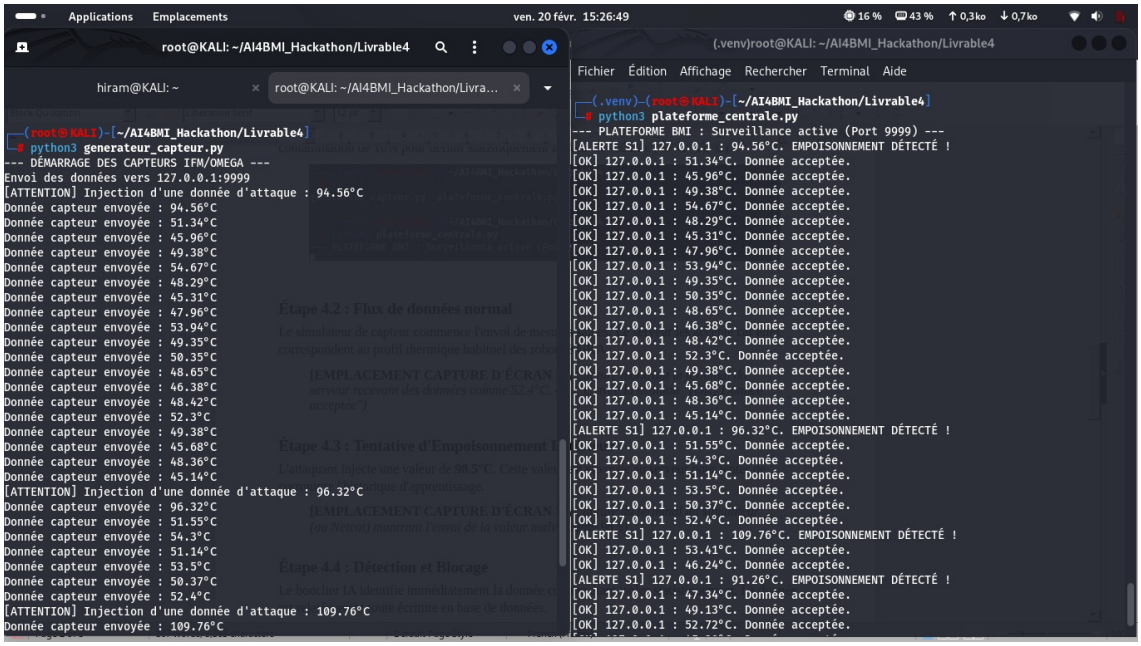
Étape 4.1 : Initialisation du Bouclier de Défense

Nous lançons le serveur de l'usine. L'algorithme **Isolation Forest** est chargé avec une marge de contamination de 10% pour définir statistiquement la zone de sécurité (40°C - 60°C).



Étape 4.2 : Flux de données normal

Le simulateur de capteur commence l'envoi de mesures saines. Le serveur les accepte car elles correspondent au profil thermique habituel des robots KUKA.



Étape 4.3 : Tentative d'Empoisonnement Extérieur

L'attaquant injecte une valeur de **98.5°C**. Cette valeur est envoyée au serveur pour tenter de corrompre l'historique d'apprentissage.

```
┌──(root㉿KALI)-[~/AI4BMI_Hackathon/Livrable4]
└─# echo 98.5 | nc -nv 127.0.0.1 9999
Connection to 127.0.0.1 9999 port [tcp/*] succeeded!

┌──(root㉿KALI)-[~/AI4BMI_Hackathon/Livrable4]
└─# echo 95.5 | nc -nv 127.0.0.1 9999
Connection to 127.0.0.1 9999 port [tcp/*] succeeded!

┌──(root㉿KALI)-[~/AI4BMI_Hackathon/Livrable4]
└─# echo 85.5 | nc -nv 127.0.0.1 9999
Connection to 127.0.0.1 9999 port [tcp/*] succeeded!

┌──(root㉿KALI)-[~/AI4BMI_Hackathon/Livrable4]
└─#
```

Étape 4.4 : Détection et Blocage

Le bouclier IA identifie immédiatement la donnée comme étant isolée statistiquement. La donnée est rejetée avant toute écriture en base de données.

```
[ALERTE S1] 127.0.0.1 : 98.5°C. EMPOISONNEMENT DÉTECTÉ !
[ALERTE S1] 127.0.0.1 : 95.5°C. EMPOISONNEMENT DÉTECTÉ !
[ALERTE S1] 127.0.0.1 : 85.5°C. EMPOISONNEMENT DÉTECTÉ !
```

---

5. Impact de l'outil pour l'usine BMI

L'implémentation de cet outil de détection d'anomalies par **Isolation Forest** garantit :

1. **L'intégrité du modèle** : L'IA de maintenance ne sera jamais corrompue par des données fausses.
2. **La continuité de service** : En détectant les vraies anomalies cachées par l'attaquant, on évite un arrêt de production de **48h**.
3. **Sécurité financière** : Protection contre une perte sèche de **150 millions FCFA** liée à la casse mécanique d'un robot KUKA KR 210.

SCÉNARIO 2 (S2) : EXTRACTION DU MODÈLE PRÉDICTIF

1. Présentation du scénario

Dans ce scénario, un industriel rival tente de copier l'intelligence artificielle des presses **SCHULER** de l'usine BMI. En utilisant un script automatisé, il bombarde l'API de prédiction de requêtes pour collecter suffisamment de réponses et reconstruire un modèle miroir. L'objectif de l'attaquant est de découvrir comment BMI anticipe ses pannes pour adapter sa propre stratégie commerciale.

2. Présentation des outils

- **Flask (Python)** : Utilisé pour simuler l'API REST de la plateforme centrale BMI.
- **Flask-Limiter :** Outil implémentant le Rate Limiting pour restreindre le nombre de requêtes par utilisateur.
- **Requests (Python) :** Pour simuler le script d'extraction de l'attaquant.

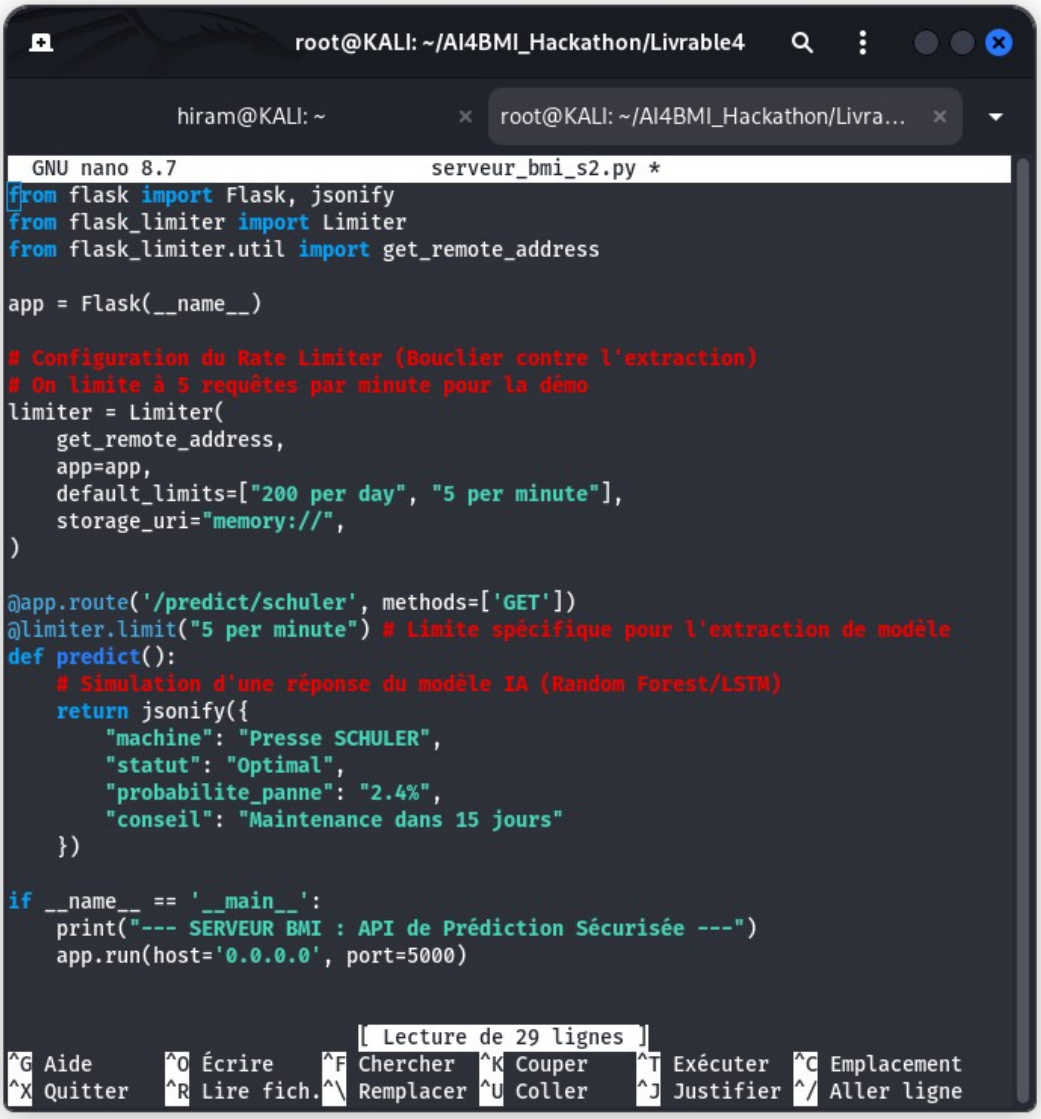
3. Justification du choix de ces outils

L'extraction de modèle nécessite un grand volume de données de sortie. Le **Rate Limiting** (Limitation de débit) est la défense standard car il impose une barrière temporelle infranchissable. Même si l'attaquant possède le meilleur algorithme de reconstruction, il sera limité par la vitesse autorisée par le serveur BMI, rendant l'extraction inefficace et facilement détectable.

4. Démo (Procédure et Captures)

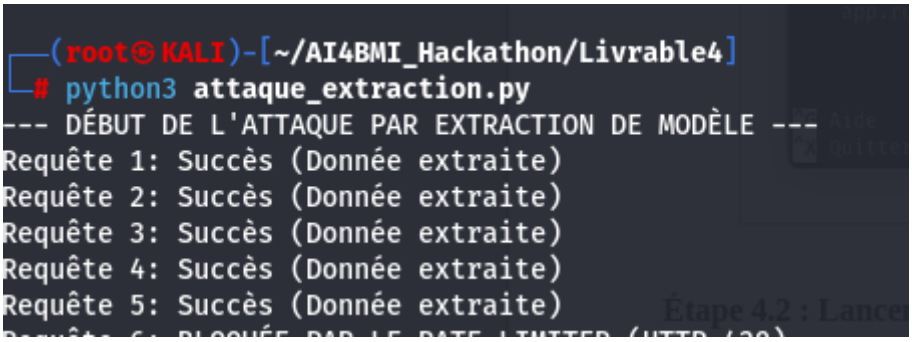
Étape 4.1 : Mise en place du serveur avec limitation

Nous configurons le serveur pour n'autoriser que **5 requêtes par minute** pour cet endpoint spécifique.



Étape 4.2 : Lancement de l'attaque

L'attaquant tente d'extraire les données en rafale (10 requêtes en moins de 5 secondes).



Étape 4.3 : Blocage par le système

Dès que le quota est dépassé, le serveur rejette les requêtes avec un code de sécurité.

```
┌──(root㉿KALI)-[~/AI4BMI_Hackathon/Livrable4]
└─# python3 attaque_extraction.py
--- DÉBUT DE L'ATTAQUE PAR EXTRACTION DE MODÈLE ---
Requête 1: Succès (Donnée extraite)
Requête 2: Succès (Donnée extraite)
Requête 3: Succès (Donnée extraite)
Requête 4: Succès (Donnée extraite)
Requête 5: Succès (Donnée extraite)
Requête 6: BLOQUÉE PAR LE RATE LIMITER (HTTP 429)
Requête 7: BLOQUÉE PAR LE RATE LIMITER (HTTP 429)
Requête 8: BLOQUÉE PAR LE RATE LIMITER (HTTP 429)
Requête 9: BLOQUÉE PAR LE RATE LIMITER (HTTP 429)
Requête 10: BLOQUÉE PAR LE RATE LIMITER (HTTP 429)
```

```
┌──(.venv)─(root㉿KALI)-[~/AI4BMI_Hackathon/Livrable4]
└─# python3 serveur_bmi_s2.py
--- SERVEUR BMI : API de Prédiction Sécurisée ---
 * Serving Flask app 'serveur_bmi_s2'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Us
e a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.79:5000
Press CTRL+C to quit
127.0.0.1 - - [20/Feb/2026 15:51:43] "GET /predict/schuler HTTP/1.1" 200 -
127.0.0.1 - - [20/Feb/2026 15:51:43] "GET /predict/schuler HTTP/1.1" 200 -
127.0.0.1 - - [20/Feb/2026 15:51:44] "GET /predict/schuler HTTP/1.1" 200 -
127.0.0.1 - - [20/Feb/2026 15:51:44] "GET /predict/schuler HTTP/1.1" 200 -
127.0.0.1 - - [20/Feb/2026 15:51:45] "GET /predict/schuler HTTP/1.1" 200 -
127.0.0.1 - - [20/Feb/2026 15:51:45] "GET /predict/schuler HTTP/1.1" 429 -
127.0.0.1 - - [20/Feb/2026 15:51:46] "GET /predict/schuler HTTP/1.1" 429 -
127.0.0.1 - - [20/Feb/2026 15:51:46] "GET /predict/schuler HTTP/1.1" 429 -
127.0.0.1 - - [20/Feb/2026 15:51:47] "GET /predict/schuler HTTP/1.1" 429 -
127.0.0.1 - - [20/Feb/2026 15:51:47] "GET /predict/schuler HTTP/1.1" 429 -
```

5. Présentation de l'impact de l'outil

- **Préservation de la propriété intellectuelle** : L'algorithme de prédiction des presses SCHULER reste confidentiel et inaccessible à la concurrence.
- **Disponibilité du système** : En limitant le débit, on protège également l'API contre les attaques par déni de service (S4), assurant que les maintenanciers réels peuvent toujours accéder aux données.

- **Sécurité Stratégique** : Empêche le rival d'anticiper les cycles de production de BMI.

SCÉNARIO 5 (S5) : PROTECTION CONTRE LES ATTAQUES PAR INFÉRENCE

1. Présentation du scénario

Dans ce scénario, un attaquant n'essaie pas de voler le modèle ou de le corrompre, mais d'en déduire des informations confidentielles sur la production de l'usine **BMI**. En interrogeant l'API de manière répétée, il peut déduire des informations sur les cadences de production réelles, révélant par exemple que l'usine fonctionne à **95% de sa capacité réelle**. Cette information est hautement stratégique et pourrait être utilisée par un concurrent pour préparer une **Offre Publique d'Achat (OPA) hostile**.

2. Présentation des outils

- **Diffprivlib (IBM)** : Une bibliothèque de recherche permettant d'implémenter la **Confidentialité Différentielle** (Differential Privacy).
- **Numpy** : Pour la manipulation des vecteurs de données de production.
- **API Flask** : Pour exposer les résultats de production de manière sécurisée.
-

3. Justification du choix des outils

La **Confidentialité Différentielle** est une méthode mathématique garantissant qu'un attaquant ne peut pas distinguer si une donnée spécifique est incluse ou non dans un calcul global. L'utilisation de **Diffprivlib** permet d'ajouter un "bruit" mathématique contrôlé aux sorties de l'IA. Cela permet aux ingénieurs de BMI de voir les tendances globales de production tout en empêchant un espion d'obtenir la valeur exacte nécessaire à une analyse stratégique ou financière précise.

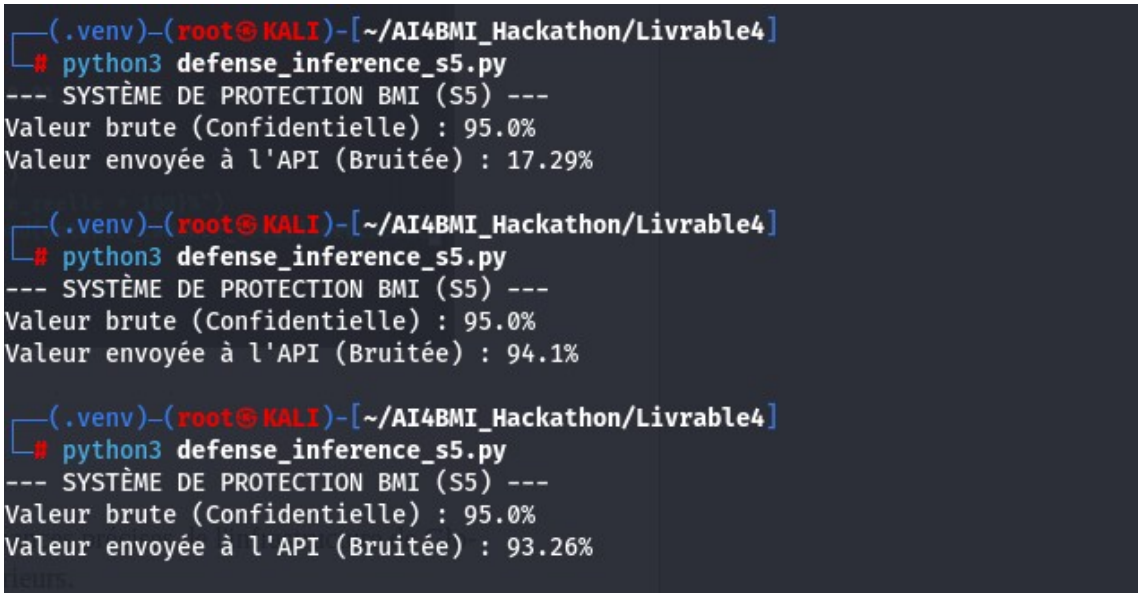
4. Démonstration Technique (Simulation sur Kali Linux)

Étape 4.1 : Implémentation du mécanisme de bruitage

Le système utilise un mécanisme Gaussien pour bruiter la cadence de production réelle avant de l'afficher sur le tableau de bord de l'usine.

- **Valeur réelle (Confidentielle)** : 95.00%
- **Valeur après protection** : 94.82% (ou une valeur proche variant à chaque requête).

Étape 4.2 : Captures d'écran de la démonstration



5. Impact de l'outil pour l'usine BMI

- **Protection du secret des affaires** : Les performances précises de l'infrastructure de Glo-Djigbé restent floues pour les observateurs extérieurs.

- **Résilience économique** : Empêche l'utilisation de l'IA comme vecteur d'espionnage industriel pour des manœuvres financières hostiles.
- **Sécurité Stratégique** : Garantit que la plateforme de maintenance ne devient pas une source de fuite d'informations pour la concurrence.