

LIVRABLE 2

L'objectif de ce livrable est de garantir la confidentialité et l'intégrité des données critiques de maintenance et des modèles d'IA. Nous avons identifié deux risques majeurs : l'espionnage industriel (fuite de données) et le sabotage opérationnel (ransomware).

Scénario 3

1. Présentation du scénario

Dans ce scénario, des hackers exploitent une faille dans l'interface d'administration et réussissent à télécharger toute la base contenant l'historique de maintenance.

Ces informations sont très sensibles car elles permettent de savoir :

- quelles machines tombent le plus souvent en panne
- à quelle fréquence
- quelles pièces sont utilisées

Les concurrents peuvent alors exploiter ces informations contre l'entreprise.

Le problème principal est donc la protection des données stockées et des communications.

2. Présentation des outils

Pour réduire ce risque, nous avons utilisé les protections suivantes.

- Pour chiffrer les données au repos, nous avons opté pour l'extension pgcrypto couplé à l'algorithme AES-256
- Pour chiffrer le trafic web nous avons utilisé **HTTPS / TLS 1.3**
- Pour les connexions distante, notre choix s'est porté sur tailscale

3. Justification du choix

pgcrypto / AES-256

Permet de chiffrer directement les données dans la base PostgreSQL.

Même si un attaquant vole la base :

- les données restent illisibles
- il faut la clé de chiffrement pour les lire.

AES-256 est un standard de l'industrie et est considéré comme très sécurisé.

HTTPS / TLS 1.3

Protège les communications entre les services. Sans TLS les données circulent en clair alors qu'avec TLS elles sont chiffrées.

Tailscale

Tailscale est un VPN moderne.

Il permet :

- de sécuriser les connexions vers les serveurs
- de créer un tunnel chiffré entre machines.

Cela empêche un attaquant présent sur le réseau d'intercepter les données.

Tailscale est choisi car il est :

- rapide
- simple à configurer
- très sécurisé.

4. Démo

Pour des raisons de compatibilité et de facilité à effectuer les tests, nous avons utilisé des conteneurs docker pour le projet.

L'environnement est constitué de 4 machines :

- mqtt (utilisé pour créer les canaux d'écoute avec et sans tls)
- pg (le serveur de base de donnée postgresql)
- sniffer (une machine qui vas capturer le trafic)
- tailscale (une machine ou est configuré le vpn tailscale)

Le fichier « docker-compose.yml » utilisé pour les tests seront dans une archive zip.

Note: Les tests avec le vpn utilise un autre fichier « docker-compose_vpn.yml » compte tenu du fait qu'il faillait faire du port forwarding.

Compte tenu du fait que nous avons utilisé un token pour pouvoir paramétrier tailscale, le fichier « docker-compose_vpn.yml » sera fourni mais pas le token.

Données chiffrés au repos

- Connexion à la base de donnée avec :

```
docker exec -it pg psql -U demo -d maintenance
```

- Affichage de la basé de donné avec :

```
SELECT * FROM maintenance_logs;
```

```

id |          data
---+-----
1 | \xc30d04070302346222937b51039a7fd270012a93ec8e77f1e10b9e313b9b575da81496e57ed6c70e0c3e809d31e5c0c6
79f39e82d3b6bf8c818fb74c248b4f338ea6f4033f4067723f9a8dfee73acce1976d97e0ff7e73dd171ac44d636d954a7b6a1b2e
fe3815e4e0eb24a59d826b7304a2d7f7897d5a76a0804bfb2007320dc
2 | \xc30d04070302c45a9285bf23ab8a7ad277012bec7c3d96dfb6c0ef335968ecb805b31999d9c0773eaf1ceefc257df192
000f266b23f31469297d710c29dc11a52a2f347110d14f4732f088e7091438e36f82002a5213e47429add7c540a2dc751cbc44c7
d1505000338de3db829c57d854d6bd2ada5d21d8d14319c1cd234f58e5179917a6b34164
3 | \xc30d040703024288fe2687b2f6787dd27901aa15b3f3570ded3e0eec35564d621111eabc317d32f4ad8b11bbd24d7c9c
905df42cbf9d4063f86987392305fd05cd319b603010f77b89bff2f161ac815a4eff12f39b6f06c67a15d7b53549d1ebbc7fc53a
66c1bbeff870af649627a5c7468671b18ef958ea2d0b685e358139022fc38370ef81abb4d3ab7
(3 rows)

```

On voit que la base de donné est chiffré avec pgcrypto couplé à AES-256.

- Déchiffrement de la base de donné avec la clé secrète

```
SELECT pgp_sym_decrypt(data, 'cle-secrete') AS message FROM maintenance_logs;
```

```

maintenance=# 
maintenance=# SELECT pgp_sym_decrypt(data, 'cle-secrete') AS message FROM maintenance_logs;
               message
-----
 {"machine":"Machine A", "panne":"Moteur HS", "date":"2026-02-22"}
 {"machine":"Machine B", "panne":"Courroie cassée", "date":"2026-02-22"}
 {"machine":"Machine C", "panne":"Sonde défectueuse", "date":"2026-02-22"}
(3 rows)

```

La base de donné est déchiffré.

Démo de chiffrement des communications avec TLS et sans TLS

- Sans TLS

D'abord on commence par capturer le trafic avec tcpdump depuis la machine « sniffer » et on le sauvegarde dans un fichier

```
docker exec -it sniffer tcpdump -i any port 1883 -w mqtt.cap
```

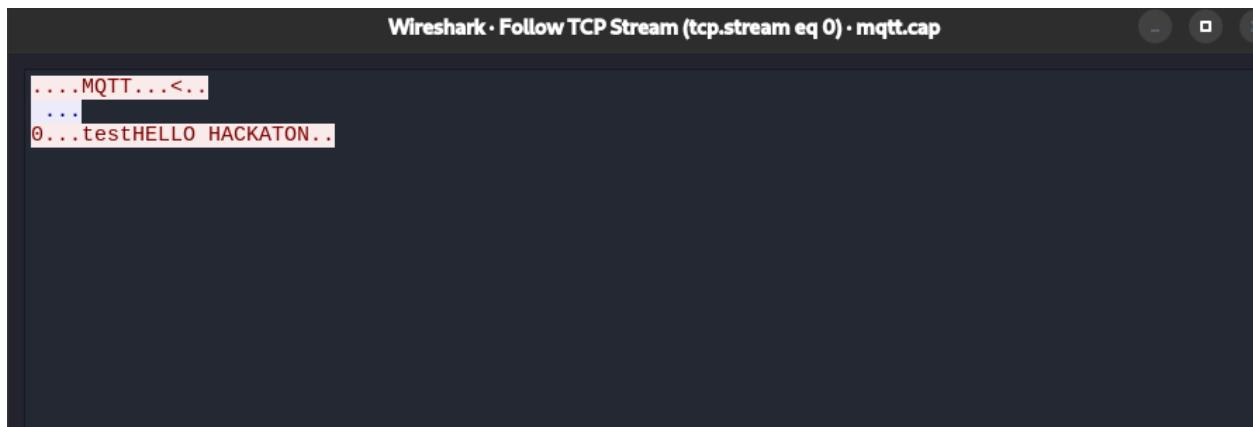
Ensute on crée un canal d'écoute sans TLS appelé « test » sur le server « mqtt »

```
docker exec -it mqtt mosquitto_sub -h localhost -t test
```

On envoie un message sur le canal non sécurisé depuis le serveur de base de donnée « pg »

```
docker exec -it pg mosquitto_pub -h mqtt -t test -m "HELLO HACKATON"
```

L'ouverture du fichier de capture dans wireshark montre le message envoyé en clair par le serveur « pg »



- Avec TLS

La commande pour générer les certificats dans le fichier « mqtt »

```
openssl genrsa -out ca.key 2048 ;  
openssl req -x509 -new -nodes -key ca.key -sha256 -days 365 -out ca.crt -subj  
"/CN=MQTT-CA" ;  
  
openssl genrsa -out server.key 2048 ;  
openssl req -new -key server.key -out server.csr -subj "/CN=localhost" ;  
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out  
server.crt -days 365 -sha256 ;
```

On capture également le trafic avec tcpdump depuis la machine « sniffer » et on le sauvegarde dans un fichier

```
docker exec -it sniffer tcpdump -i any port 8883 -w mqtt.cap
```

Ensuite on crée le canal d'écoute sécurisé avec TLS sur le serveur « mqtt »

```
docker exec -it mqtt sh -c 'mosquitto_sub -h localhost -p 8883 --cafile  
/mosquitto/certs/ca.crt -t test'
```

Et on envoie un message sur ce canal depuis le serveur de base de donnée « pg »

```
docker exec -it pg sh -c 'mosquitto_pub -h mqtt -p 8883 --cafile /root/ca.crt -t test -m  
"HELLO HACKATON" --insecure'
```

Le fichier de capture wireshark montre que les données sont chiffré et donc illisible.



Démo avec le VPN tailscale

D'abord on a créé un compte sur « tailscale.com » et à l'aide d'un token on expose notre machine (ici le conteneur tailscale) au travers de

tailscale.com. Les services sont exposé via tailscale à l'adresse IP attribué.

Machines

Manage the devices connected to your tailnet. [Learn more](#)

Add device

Search by name, owner, tag, version...

Filters

3 machines

MACHINE	ADRESSES ⓘ	VERSION	LAST SEEN
hackathon-server ...@gmail.com	10.0.0.4	1.94.2 Linux 6.18.9+kali-amd64	Connected

On se connecte à tailscale avec la commande « sudo tailscale login » et en suivant le lien.

```
~/Desktop/Hackaton/demo
> sudo tailscale login
[sudo] password for ir0nx:

To authenticate, visit:

    https://login.tailscale.com/a/15b532c90183db

Success.
```

On voit sur le dashboard qu'on est connecté au réseau VPN



En utilisant l'adresse de « hackaton-server » et les ports associés on peut se connecter à la base de données MySQL.

```
~/Desktop/Hackaton/demo
> psql -h [REDACTED] -U demo -d maintenance
Password for user demo:
psql (18.1 (Debian 18.1-2), server 15.16 (Debian 15.16-1.pgdg13+1))
Type "help" for help.

maintenance=# \dt
          List of tables
 Schema |      Name      | Type | Owner
-----+-----+-----+-----
 public | maintenance_logs | table | demo
(1 row)

maintenance=#
```

On voit que la connexion est établi avec succès.

Par contre une fois déconnecté du vpn, on n'as plus accès au réseau interne.

```
~/Desktop/Hackaton/demo
> sudo tailscale logout

~/Desktop/Hackaton/demo
> psql -h [REDACTED] -U demo -d maintenance
^C
```

Note : Tailscale est également disponible sur téléphone. Dans la vidéo finale sera présenté une connexion à la base de donné depuis un téléphone en utilisant le vpn.

5. Impact de l'outil

Avec ces protections :

- une base volée est inutilisable
- les communications ne peuvent pas être espionnées
- l'accès aux serveurs est sécurisé.

Cela réduit fortement le risque de fuite de données.

Scénario 4

1. Présentation du scénario

Dans ce scénario, un ransomware attaque l'entreprise.

Le malware chiffre :

- la base de données
- les modèles d'intelligence artificielle

Conséquences :

- plus de prédictions
- maintenance plus lente
- pannes en cascade
- pertes financières importantes.

La solution principale est la sauvegarde et la gestion sécurisée des clés.

2. Présentation de l'outil

- **pg_dump**

3. Justification du choix

Pg_dump

Pg_dump permet de sauvegarder et de restaurer rapidement une base de données PostgreSQL

En cas de ransomware :

- on supprime la base infectée
- on restaure la dernière sauvegarde.

Cela réduit fortement l'impact de l'attaque.

4. Démo

Sauvegarde de la base de données

```
docker exec -it pg sh -c 'pg_dump -U demo -d maintenance > /backup/backup_maintenance.sql'
```

Simulation d'un ransomware (base de données supprimé)

```
docker exec -it pg psql -U demo -d maintenance -c "DROP TABLE maintenance_logs;"
```

```
~/Desktop/Hackaton/demo
> docker exec -it pg psql -U demo -d maintenance -c "DROP TABLE maintenance_logs;"
```

Vérification de l'état de la base

```
~/Desktop/Hackaton/demo
> docker exec -it pg psql -U demo -d maintenance
psql (15.16 (Debian 15.16-1.pgdg13+1))
Type "help" for help.

maintenance=# SELECT * FROM maintenance_logs;
ERROR: relation "maintenance_logs" does not exist
LINE 1: SELECT * FROM maintenance_logs;
          ^
maintenance=#

```

On voit que n'existe plus.

Restauration de la base

```
docker exec -ti pg sh -c 'psql -U demo -d maintenance <
/backup/backup_maintenance.sql'
```

```
~/Desktop/Hackaton/demo
> docker exec -ti pg sh -c 'psql -U demo -d maintenance < /backup/backup_maintenance.sql'
SET
SET
SET
SET
SET
SET
set_config
-----
(1 row)

SET
SET
SET
SET
CREATE EXTENSION
COMMENT
SET
SET
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER TABLE
ALTER SEQUENCE
ALTER TABLE
COPY 3
  setval
-----
      3
(1 row)

ALTER TABLE
```

On voit que la base de données est restaurée et toujours chiffré et disponible.

5. Impact de l'outil

Grâce à cet outil :

- l'entreprise peut récupérer rapidement après une attaque par ransomware
- les dégâts d'un ransomware sont limités.

Cela améliore fortement la résilience du système.