Université Moulay Ismaïl
Faculté des Sciences et Techniques d'Errachidia

Cycle d'Ingénieur en Génie Informatique et Business Intelligence

Module : Cryptographie et sécurité informatique

Mini Projet : DS2OS Anomaly Detection

Réalisé par :

Youssef Tinid

Encadré par :

Prof. **Fatima Amounas**
Tuteur Pédagogique

Année Universitaire : 2025–2026

Table des matières

1. Contexte et Problématique

1.1 Problématique

Avec le développement rapide de l'Internet des objets (IoT), les réseaux génèrent un trafic massif et hétérogène provenant de multiples capteurs et dispositifs intelligents. Ces flux contiennent des informations critiques, et leur sécurité est essentielle pour le bon fonctionnement des environnements sensibles tels que la santé, l'industrie ou les maisons intelligentes.

Cependant, certaines situations peuvent compromettre la fiabilité du système :

— Les anomalies dans les flux IoT peuvent représenter des **comportements malveillants** ou des **erreurs système**.

— Les intrusions ou accès non autorisés peuvent affecter la disponibilité et la sécurité des données.

— Le trafic IoT évolue dans le temps, entraînant des changements dans le comportement normal des appareils (**concept drift**), ce qui nécessite des modèles adaptatifs.

Ainsi, l'objectif est de concevoir un modèle capable de **détecter automatiquement les anomalies** tout en s'adaptant aux variations du trafic IoT.

1.2 Dataset DS2OS

Pour réaliser cette tâche, le projet utilise le dataset **DS2OS IoT Traffic Traces** :

— **Source** : Traces de trafic IoT collectées à partir de quatre sites IoT simulés, comprenant plusieurs types de services et dispositifs.

— **Types d'appareils** : Contrôleurs de lumière, thermostats, portes intelligentes, capteurs de mouvement, capteurs de batterie, etc.

— **Labels** : Binaire — *normal* (0) ou *anomalie* (1), permettant d'évaluer les performances des modèles de détection.

— **Caractéristiques** :

  — Attributs liés aux types d'appareils et à leur localisation.

  — Opérations réalisées sur les dispositifs (lecture, écriture, verrouillage, abonnement aux services).

  — Valeurs sémantiques et mesures spécifiques des dispositifs.

  — Caractéristiques temporelles et statistiques (fréquence des événements, heures, delta-time, etc.).

Ce dataset est particulièrement adapté pour évaluer des **modèles adaptatifs de détection d'anomalies** dans des environnements IoT simulés.

2. Méthodologie

2.1 Analyse exploratoire des données initiales

Le jeu de données DS2OS contient **357 952 enregistrements** et **13 colonnes**, décrivant le trafic IoT au niveau applicatif. Les 13 colonnes sont les suivantes :

— **sourceID** : identifiant unique de l'appareil source (84 valeurs uniques, ex : `lightcontrol1`, `movement2`, `tempin4`)

— **sourceAddress** : adresse du service source (89 valeurs uniques)
— **sourceType** : type de l'appareil source (8 valeurs uniques, ex : `/lightControler`, `/movementSensor`)
— **sourceLocation** : emplacement physique de l'appareil source (21 valeurs uniques, ex : `BedroomParents`, `Kitchen`)
— **destinationServiceAddress** : adresse du service cible (85 valeurs uniques)
— **destinationServiceType** : type du service cible (8 valeurs uniques)
— **destinationLocation** : emplacement physique du service cible (21 valeurs uniques)
— **accessedNodeAddress** : adresse du noeud accédé (170 valeurs uniques)
— **accessedNodeType** : type du noeud accédé (12 valeurs uniques, ex : `/lightControler`, `/derived/boolean`)
— **operation** : type d'opération effectuée (5 valeurs uniques, ex : `registerService`, `write`, `read`)
— **value** : valeur de l'opération ou mesure (10 623 valeurs uniques, incluant des nombres, `none` et timestamps)
— **timestamp** : horodatage de l'événement (format UNIX)
— **normality** : étiquette binaire indiquant un comportement normal (0) ou anomalie (1)

Cette section permet de comprendre la nature des données avant toute étape de prétraitement et d'encodage pour les modèles ML et DL.

Les types et la présence de valeurs manquantes sont résumés dans le tableau ci-dessous :

— **Types de données** : 11 colonnes catégorielles, 2 colonnes numériques (int64 et timestamp)
— **Valeurs manquantes** :
  — `accessedNodeType` : 148 valeurs manquantes (0,04 %)
  — `value` : 2 050 valeurs manquantes (0,57 %)
— **Doublons** : 1 912 enregistrements
— **Colonnes vides** : aucune

Les statistiques numériques principales sont les suivantes :

— `timestamp` : min = 1.520032e+12, max = 1.520118e+12, moyenne = 1.520078e+12
— `normality` (cible) : 0 (normal) = 347 935 (97,2 %), 1 (anomalie) = 10 017 (2,8 %)

2.2 Analyse des colonnes catégorielles

Le jeu de données contient 11 colonnes catégorielles importantes, avec le nombre de valeurs uniques et quelques exemples :

— `sourceID` : 84 valeurs uniques, ex : `lightcontrol1`, `movement2`, `tempin4`
— `sourceAddress` : 89 valeurs uniques
— `sourceType` : 8 valeurs uniques, ex : `/lightControler`, `/movementSensor`, `/sensorService`
— `sourceLocation` : 21 valeurs uniques, ex : `BedroomParents`, `Kitchen`, `Garage`
— `destinationServiceAddress` : 85 valeurs uniques

— `destinationServiceType` : 8 valeurs uniques
— `destinationLocation` : 21 valeurs uniques
— `accessedNodeAddress` : 170 valeurs uniques
— `accessedNodeType` : 12 valeurs uniques
— `operation` : 5 valeurs uniques, ex : `registerService`, `write`, `read`
— `value` : 10 623 valeurs uniques, incluant des nombres, `none` et des timestamps

2.3 Prétraitement et Feature Engineering

Avant d'entraîner les modèles, il est essentiel de préparer les données afin de garantir leur qualité et la pertinence des caractéristiques extraites. Cette section détaille les étapes effectuées pour le prétraitement et l'ingénierie de features.

2.3.1 Suppression des doublons et traitement des valeurs manquantes

Le premier traitement consiste à nettoyer le jeu de données en supprimant les doublons et en gérant les valeurs manquantes.

Pour certaines colonnes comme `value` ou `accessedNodeType`, une colonne indicatrice (`is_nan`) a été créée pour conserver l'information de présence d'une valeur manquante.

2.3.2 Encodage des colonnes catégorielles à haute cardinalité

Les colonnes `sourceID`, `accessedNodeAddress`, `src_device_info`, `dest_device_info` et `acc_node_info` contiennent de nombreuses valeurs uniques. Pour ne pas multiplier les dimensions, un encodage par fréquence (Frequency Encoding) a été appliqué.

Cette méthode remplace chaque valeur par sa fréquence relative dans le jeu de données, permettant aux modèles de capturer la popularité d'un identifiant sans créer trop de colonnes.

2.3.3 Création de features relationnelles

Certaines caractéristiques sont construites à partir de relations entre colonnes existantes, par exemple :

— `src_dest_same_addr` : 1 si l'adresse source est identique à l'adresse de destination, 0 sinon.
— `src_dest_same_type` : identique pour le type d'appareil.
— `src_dest_same_loc` : identique pour la localisation.

2.3.4 Séparation des informations dans les chemins

Pour les colonnes contenant des chemins de type `/agent2/lightcontrol2`, nous avons séparé :

— le numéro d'agent : `src_agent_number`, `dest_agent_number`
— le type de device : `src_device_info`, `dest_device_info`, `acc_node_info`

2.3.5 Traitement de la colonne `value`

La colonne `value` contient des données mixtes (numérique, textuel, 'none'). Nous avons créé des colonnes dérivées :

— `value_numeric` : conversion en numérique, NaN si non numérique.
— `value_is_none` : 1 si `value` = 'none'.
— `value_category` : catégorisation des valeurs textuelles fréquentes et des autres.

De plus, des colonnes spécifiques ont été créées par type d'événement, par exemple `value_lightcontrol`, `value_tempin`, etc.

2.3.6 Encodage des types d'appareils, des localisations et des opérations

— **Types d'appareils** : un score de risque a été attribué à chaque type et les 3 types les plus fréquents ont été one-hot encodés.
— **Localisations** : un score de confidentialité (privacy score) a été assigné selon la pièce.
— **Opérations** : un score de criticité et un encodage one-hot pour chaque type d'opération.

2.3.7 Encodage temporel

La colonne `timestamp` a été convertie en :

— `year`, `month`, `day`, `hour`, `minute`, `second`
— transformation circulaire pour l'heure : `hour_sin`, `hour_cos` afin de conserver la continuité temporelle.
— `delta_time` : temps écoulé depuis le dernier événement pour chaque `sourceID`.
— `events_per_min` : fréquence des événements par minute.
— indicateurs de période : `is_night`, `is_work_hours`.

2.3.8 Suppression des colonnes inutiles

Après la création des nouvelles caractéristiques et l'encodage des colonnes catégorielles, certaines colonnes originales ne sont plus nécessaires pour l'apprentissage, car :

— elles sont redondantes avec les nouvelles features (ex : `sourceID` et `sourceID_freq`)
— elles contiennent des chaînes de caractères non exploitables directement par les modèles (ex : `sourceAddress`, `destinationServiceAddress`)
— elles sont obsolètes après transformation (ex : `value_numeric` et `value_category` remplacent `value`)
— la colonne `value_lastChange` a été supprimée après analyse : certaines différences de temps (`delta_lastChange_s`) étaient négatives, et après investigation, tous les appareils possédant cette colonne avaient `normality = 0`. Cette colonne n'apporte donc aucune information discriminante pour la détection d'anomalies.
— certaines colonnes temporelles détaillées (`year`, `month`, `day`, `hour`, ...) sont transformées en features utiles (`hour_sin`, `hour_cos`, `delta_time`)

Après cette étape, le jeu de données contient uniquement les colonnes pertinentes pour l'apprentissage automatique, réduisant la dimensionnalité et améliorant l'efficacité du modèle.

2.3.9 Standardisation des colonnes numériques

Les colonnes numériques comme `value_tempin`, `value_charge`, `delta_time` et `events_per_min` ont été standardisées pour faciliter l'entraînement des modèles DNN et ML.

2.4 Évaluation et visualisation des performances

Cette section présente l'évaluation des modèles de détection d'anomalies entraînés, à savoir :

— Random Forest,

— Support Vector Machine (SVM),

— Réseau de neurones profond (DNN).

L'évaluation repose sur plusieurs métriques classiques de classification binaire ainsi que sur des visualisations permettant d'analyser le comportement de chaque modèle.

2.4.1 Évaluation du modèle Random Forest

Le modèle Random Forest a été évalué sur l'ensemble de test à l'aide d'une matrice de confusion et d'un rapport de classification.

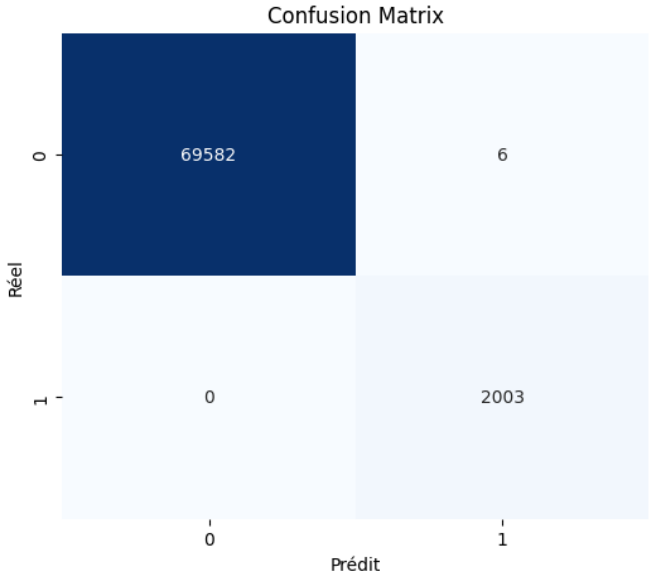

FIGURE 1 – Matrice de confusion du modèle Random Forest

La matrice de confusion montre que le modèle Random Forest détecte correctement la quasi-totalité des flux normaux et anormaux. On observe :

— un très faible nombre de faux positifs,

— aucune anomalie manquée (faux négatifs),

— une séparation nette entre les classes normales et anormales.

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     69588
           1       1.00      1.00      1.00      2003

    accuracy                           1.00     71591
   macro avg       1.00      1.00      1.00     71591
weighted avg       1.00      1.00      1.00     71591

ROC-AUC: 0.9999999569536884
PR-AUC: 0.9999985052359096
```

FIGURE 2 – Rapport de classification du modèle Random Forest

Le rapport de classification confirme les excellentes performances du modèle, avec des valeurs proches de 1 pour la précision, le rappel et le F1-score. Les scores ROC-AUC et PR-AUC proches de 1 indiquent une très forte capacité de discrimination entre trafic normal et trafic anormal.

2.4.2 Évaluation du modèle SVM

Le modèle SVM a également été évalué sur les mêmes données afin de comparer ses performances avec celles du Random Forest.

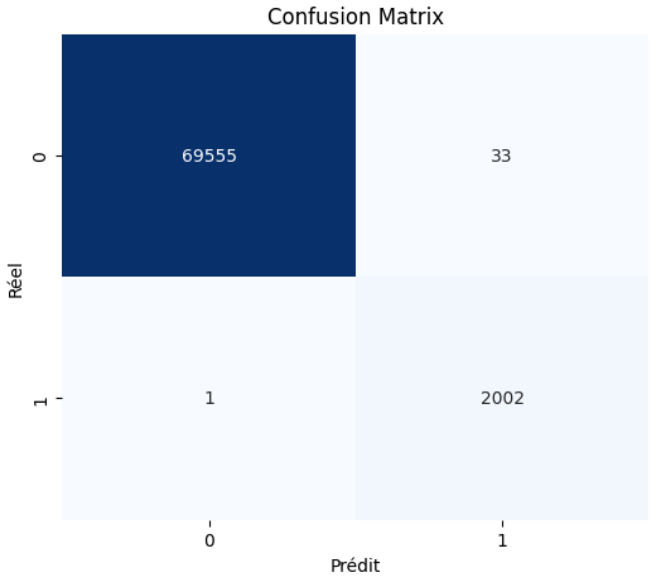

FIGURE 3 – Matrice de confusion du modèle SVM

La matrice de confusion du SVM montre une très bonne détection globale, bien que le nombre de faux positifs et de faux négatifs soit légèrement supérieur à celui du Random Forest. Cela indique une sensibilité légèrement moindre face à certaines anomalies spécifiques.

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     69588
           1       0.98      1.00      0.99      2003

    accuracy                           1.00     71591
   macro avg       0.99      1.00      1.00     71591
weighted avg       1.00      1.00      1.00     71591

ROC-AUC: 0.9997947085522202
PR-AUC: 0.9802769900167054
```

FIGURE 4 – Rapport de classification du modèle SVM

Les métriques de performance du SVM restent élevées, avec un F1-score proche de 1. Cependant, la légère baisse du PR-AUC par rapport au Random Forest suggère une sensibilité accrue aux faux positifs lorsque les classes sont déséquilibrées.

2.4.3 Évaluation du modèle DNN

Un modèle de Deep Neural Network a été entraîné afin d'exploiter la capacité des réseaux neuronaux à apprendre des représentations complexes des données.

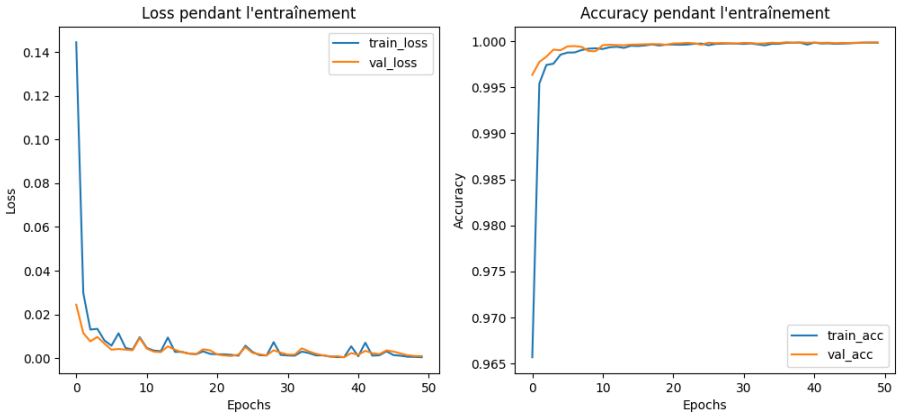

FIGURE 5 – Évolution de la loss et de l'accuracy du modèle DNN

Les courbes d'apprentissage montrent une convergence rapide du modèle. La diminution progressive de la loss ainsi que l'augmentation stable de l'accuracy indiquent un apprentissage efficace sans signe notable de sur-apprentissage.

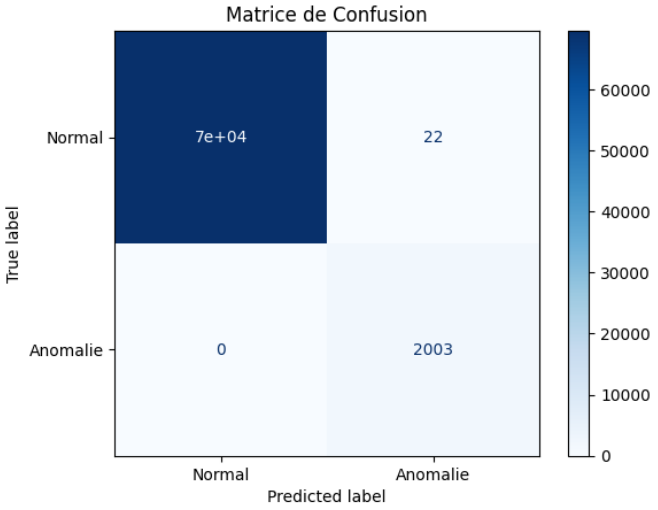

FIGURE 6 – Matrice de confusion du modèle DNN

La matrice de confusion du DNN montre que toutes les anomalies ont été correctement détectées. Le nombre de faux positifs reste limité, ce qui traduit une bonne capacité de généralisation du modèle.

```
Precision: 0.9891
Recall   : 1.0000
F1-Score : 0.9945
ROC curve AUC =  0.9999868816365036
```

FIGURE 7 – Métriques de performance du modèle DNN

Les métriques globales confirment les très bonnes performances du DNN avec :

— une précision élevée,

— un rappel parfait,

— un F1-score proche de 1,

— une aire sous la courbe ROC quasiment égale à 1.

Ces résultats montrent que le DNN est particulièrement efficace pour la détection d'anomalies dans le trafic IoT.

3. Analyse des résultats

Les résultats obtenus montrent que les trois modèles testés (Random Forest, SVM et DNN) offrent des performances très élevées pour la détection d'anomalies dans le trafic IoT DS2OS.

Le modèle **Random Forest** présente les meilleures performances globales, avec un taux d'erreur extrêmement faible, aucune anomalie manquée et des valeurs ROC-AUC et PR-AUC proches de 1. Sa robustesse face au déséquilibre des classes en fait un choix particulièrement adapté à ce type de problème.

Le **SVM** affiche également de très bonnes performances, mais reste légèrement moins précis que le Random Forest, notamment en termes de faux positifs et de PR-AUC. Cela indique une sensibilité accrue au déséquilibre des données.

Le **DNN** démontre une excellente capacité d'apprentissage, avec une convergence rapide et un rappel parfait. Malgré un nombre légèrement supérieur de faux positifs par rapport au Random Forest, ses performances restent très compétitives et confirment l'intérêt des approches deep learning pour la détection d'anomalies complexes.

Cependant, ces résultats élevés peuvent également indiquer un possible biais lié à la nature simulée du dataset DS2OS. Une évaluation sur des données réelles ou en présence de *concept drift* plus marqué serait nécessaire pour confirmer la robustesse des modèles en conditions réelles.

4. Conclusion

Dans ce projet, nous avons étudié la détection d'anomalies dans le trafic IoT à l'aide de techniques de Machine Learning et de Deep Learning appliquées au dataset DS2OS.

Les étapes de prétraitement, d'ingénierie des caractéristiques et d'entraînement des modèles ont permis d'obtenir des performances très élevées, en particulier avec le modèle Random Forest et le réseau de neurones profond.

Les résultats montrent que les approches supervisées sont efficaces pour identifier les comportements anormaux dans des environnements IoT simulés. En perspective, ce travail pourrait être étendu vers une détection en temps réel, une meilleure gestion du *concept drift*, ainsi qu'un déploiement sur des systèmes embarqués ou edge computing.

5. Références

— Aubet, F.-X., & Pahl, C. (2018). *Machine Learning-Based Adaptive Anomaly Detection in Smart Spaces*. IEEE.

— DS2OS Traffic Traces Dataset, Kaggle.

— Scikit-learn Documentation : `https://scikit-learn.org`

— TensorFlow Documentation : `https://www.tensorflow.org`