Note intermédiaire rapport de stage

Mathieu Thomassin

2024-07-31

L'Insee a renforcé récemment ses capacités opérationnelles en terme de cybersécurité via notamment la création de son SOC (Security Operation Center) en septembre 2023. Devant la quantité de données et leur diversité et face aux évolutions des techniques et tactiques des attaquants, les méthodes de détection de cyberattaques peuvent avoir des limites. L'application d'algorithmes de Machine Learning peut aider les analystes SOC à repérer des attaques. Se déroulant au sein de l'équipe SOC construite depuis peu, le stage va permettre d'appliquer des techniques de Machine Learning et de deep learning sur des jeux de données réelles, afin de participer à la détection d'incidents de sécurité.

Table of contents

1 Introduction

- **Présentation du stagiaire et du maître de stage** :

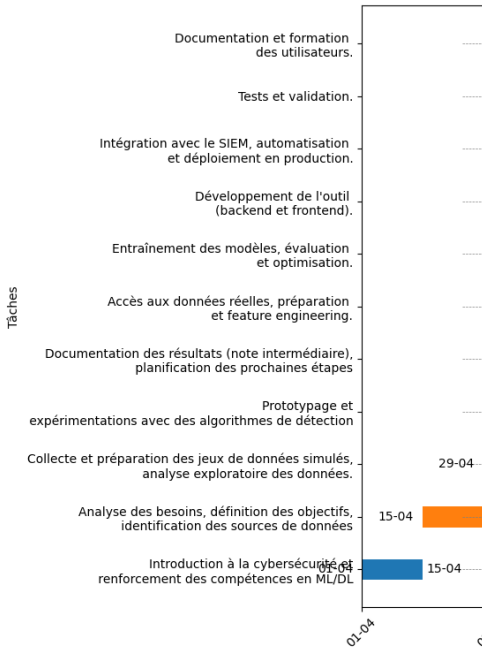
- stagiaire: Mathieu THOMASSIN
- maître de stage: Michael ORSUCCI
- Titre du stage: Machine Learning appliqué à la cybersécurité

2 Description de la mission

- **Environnement de travail**:
  - *Présentation de l'organisation* : L'Insee a renforcé récemment ses capacités opérationnelles en terme de cybersécurité via notamment la création de son SOC (Security Operation Center) en septembre 2023. Cette équipe est répartie entre les sites de la DR de Nantes et la DR de Metz et a pour objectif de renforcer la sécurité du SI.
  - *Contexte général du stage*: Devant la quantité de données et leur diversité et face aux évolutions des techniques et tactiques des attaquants, les méthodes de détection de cyberattaques peuvent avoir des limites. L'application d'algorithmes de Machine Learning peut aider les analystes SOC à repérer des attaques. Le stage va permettre d'appliquer des techniques de Machine Learning et de deep learning sur des jeux de données réelles, afin de participer à la détection d'incidents de sécurité.
- **Objectifs du Stage** :
  - *Objectifs principaux*: Il s'agit d'appliquer des techniques de détection de requêtes malveillantes arrivant dans le SI de l'Insee. Les requêtes arrivant au sein du SI peuvent être centralisées par un SIEM (Security information and event management).
  - *Résultats attendus*:
    - Offrir un service s'ajoutant au SIEM permettant d'identifier des requêtes malveillantes.
    - Pouvoir comparer différents *meilleurs* modèles (au sens d'une recherche dans les hyper paramètres) entre eux
    - Favoriser les bonnes pratiques du MLOps: reproductibilité, contrôle de version, automatisation, surveillance, collaboration
    - Étendre la méthodologie à des jeux de données publics différents
    - Explorer les algorithmes de détection d'anomalie
- **Enjeux** :
  - *Importance de la mission pour l'Insee*: à titre d'exemple, l'Insee détient l'application Elire. Une attaque réussie sur cette application perturberait le bon déroulement de la vie démocratique.
  - *Impact potentiel des résultats*:
    - La détection de requêtes malveillantes pourrait permettre de déjouer une attaque en cours.

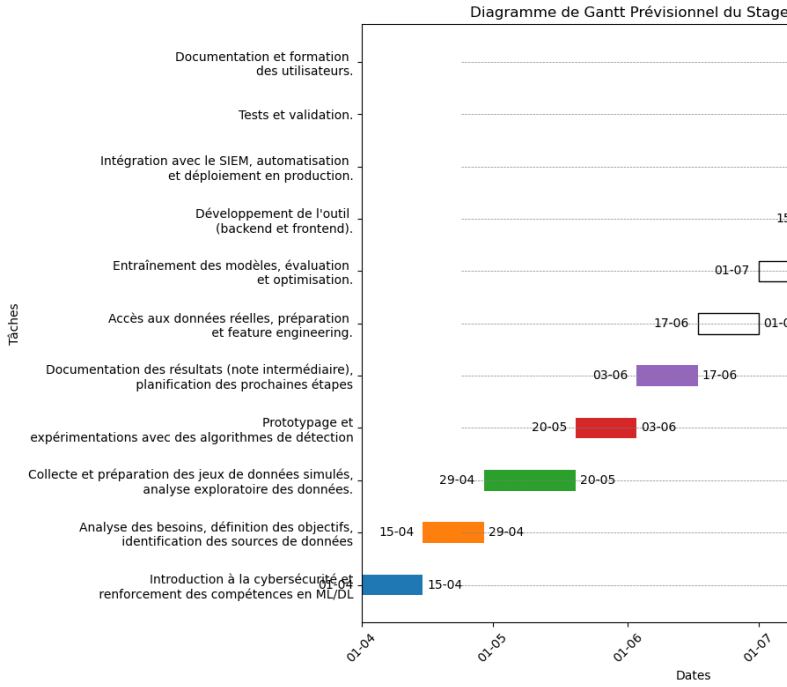
3 Timing Prévisionnel

- **Calendrier des Étapes** :



  - Détail du planning prévisionnel des différentes phases du projet
  - Échéances importantes et jalons:
    * note intermédiaire (1er juin)
    * être prêt pour tester pendant les élections européennes (8 et 9 juin)
    * surveillance pendant les jeux olympiques (26 juillet au 11 août)
    * rapport final (fin août)

- **Avancement Actuel** :



  - Comparaison avec le calendrier initial:
  - Retards éventuels et raisons:
    * Les premiers tests et expérimentations ont eu lieu, cependant, j'ai préféré commencer à comprendre plus en profondeur les design patterns de deep learning ainsi que le fonctionnement de MLFlow avant de pousser davantage la comparaison des modèles et des datasets. MLFLow va permettre une bonne méthode de comparaison et de sélection des meilleurs modèles, dans la perspective d'une intégration avec une mise en production.

    Je n'utilise donc pour le moment qu'un seul dataset constitué de bonnes et mauvaises requêtes labellisées, un seul modèle (sur plusieurs déjà entraînés).

    Je n'ai pas encore terminé l'intégration sur MLFlow.

4 Méthodologies Utilisées et Prévisionnelles

- Dépôt GitHub du code source
- **Collecte de Données** :
  - Sources des données:
    * dataset public pour l'entraînement des modèles:
      · http://www.secrepo.com/#
      · https://www.netresec.com/index.ashx?page=PcapFiles

        - https://www.stratosphereips.org/datasets-overview
        - description des datasets
        - datasets-for-network-security-e25238704c7f
        - dépôt github contenant des liens vers plusieurs datasets
        - Good et Bad queries dataset
      * données issues du SIEM, soit le POC en cours (fin mai) soit potentiellement le SIEM qui sera mis en place
  - Méthodes de collecte:
    * téléchargement à partir des liens
    * téléchargement au format Json à partir d'une recherche dans Splunk. Il serait souhaitable de réussir à utiliser l'API du SIEM afin d'automatiser cette tâche et de reproduire les résultats.

- **Préparation et Nettoyage des Données** :
  - **Lecture des fichiers de requêtes**, étiquetage en fonction de leur nature (bonne ou mauvaise) et concaténation dans un DataFrame.
  - **Étiquetage des données** : - Les données sont lues à partir de fichiers texte et chaque ligne est étiquetée comme étant soit une mauvaise requête (1) soit une bonne requête (0)
  - **Séparation des ensembles d'entraînement et de test** : - Les données étiquetées sont divisées en deux ensembles : un ensemble d'entraînement (80%) et un ensemble de test (20%).
  - **Tokenisation personnalisée** :
    * Une fonction de tokenisation personnalisée utilise des expressions régulières pour extraire des tokens pertinents des requêtes, y compris les mots, les chiffres, les URL et certains caractères spéciaux.
  - **Vectorisation TF-IDF** : - Les données textuelles sont transformées en matrices de caractéristiques en utilisant la vectorisation TF-IDF.

- **Analyse et Modélisation** :
  - **Extraction des caractéristiques** :
    * Une fonction `extract_features` est définie pour extraire les parties importantes des requêtes, en particulier les noms de scripts ou de fichiers ciblés.
    * Les données sont nettoyées pour éliminer les caractères non désirés et les retours à la ligne, puis les scripts/fichiers sont extraits et ajoutés dans une nouvelle colonne Script du DataFrame.
  - **Statistiques descriptives** :

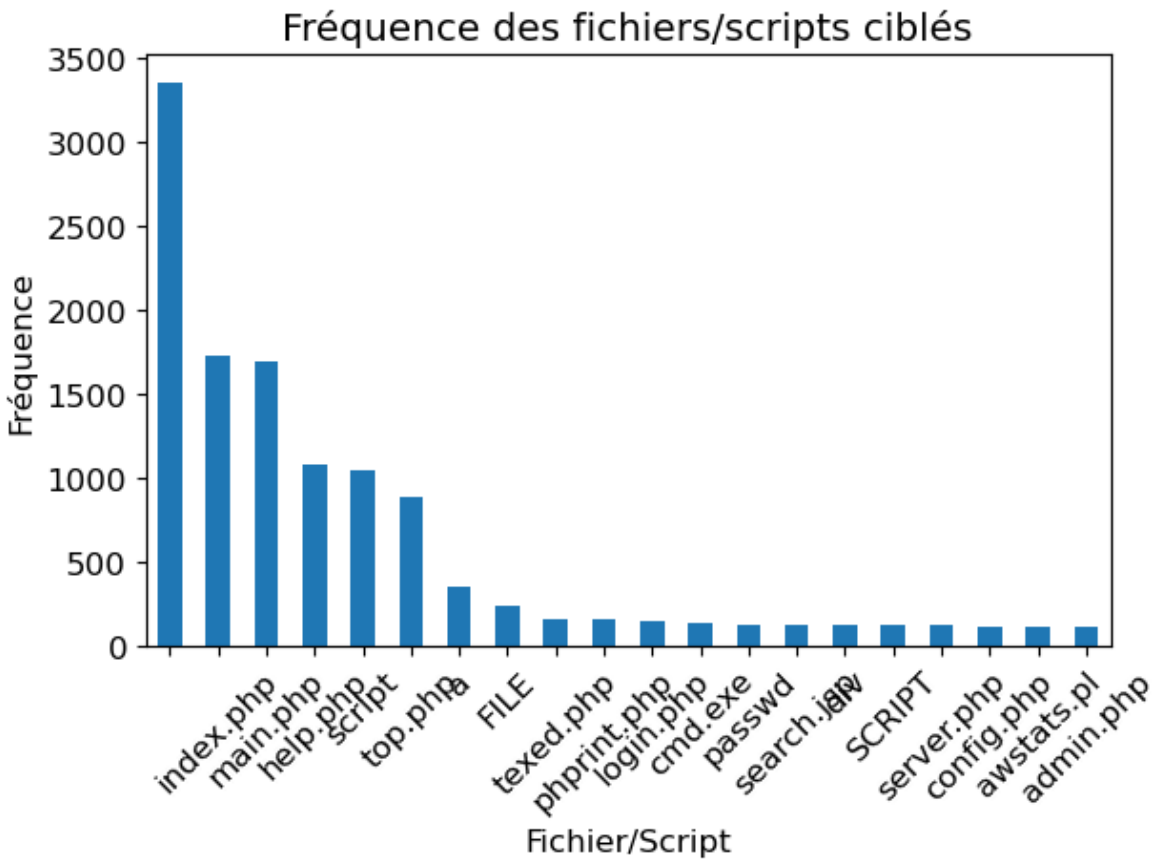


Figure 1: Les 20 scripts/fichiers les plus fréquents sont affichés

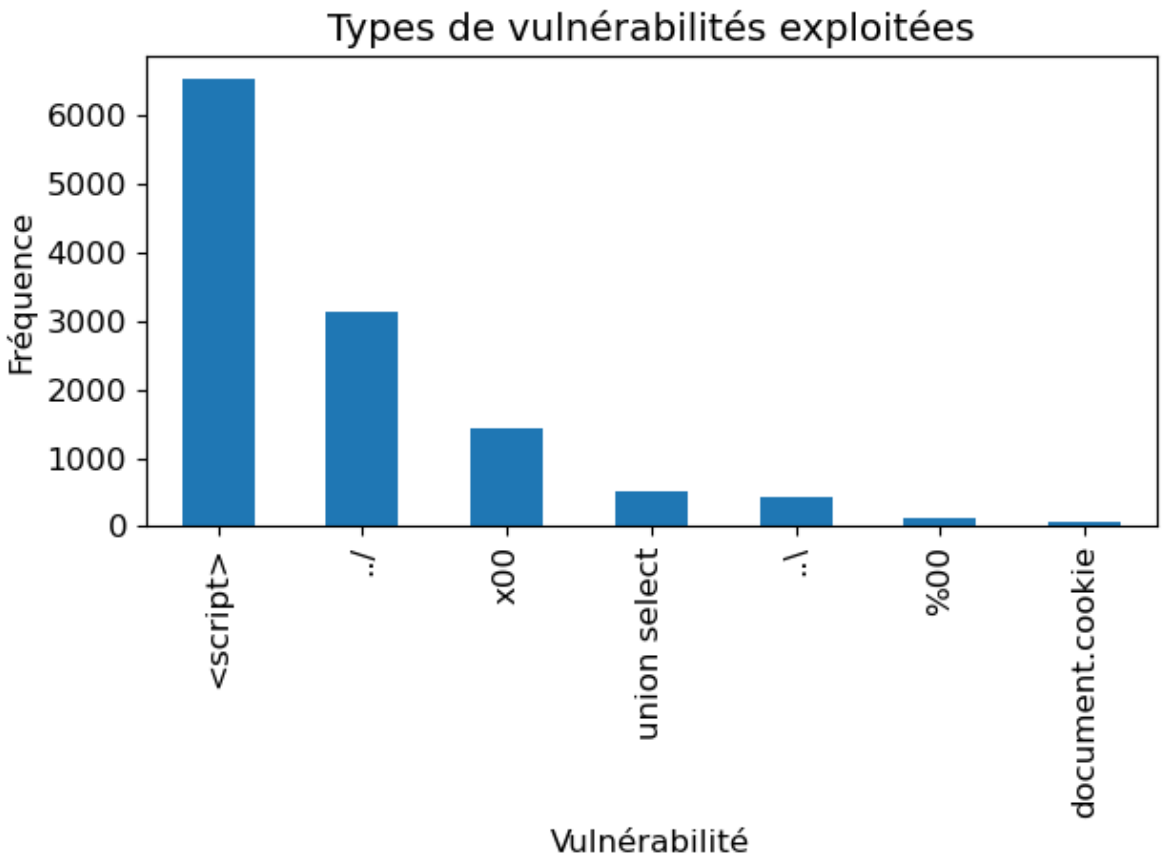


Figure 2: Types de vulnérabilités exploitées

- Modèles de machine learning envisagés:
  - régression logistique: fait et efficace à 99,6%, n'a pas révélé de requêtes malveillantes sur des données réelles issues d'une zone sécurisée.
  - K-means: envisagé mais sujet à la malédiction de la dimension. Les matrices issues de la tokenization sont *sparse*. Comment faire ?
  - SVM: pas encore tenté
  - Deep Learning:
    - un modèle CNN pour la classification de texte. Il utilise une couche d'embedding pour représenter les mots, une couche de convolution pour extraire les caractéristiques locales, une couche de pooling pour réduire la dimensionnalité et une couche dense pour effectuer la classification binaire.

      ```
      model = Sequential()
      model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim))
      model.add(Conv1D(filters=128, kernel_size=5, activation='relu'))
      model.add(GlobalMaxPooling1D())
      ```

```
model.add(Dense(1, activation='sigmoid'))
```

* Un réseau de neurones séquentiel utilisant une couche LSTM pour traiter des séquences de données et capturer des dépendances temporelles ou séquentielles. La sortie de la couche LSTM est passée à une couche dense avec une activation sigmoïde pour effectuer une classification binaire.

```
model_lstm = Sequential()
model_lstm.add(LSTM(64, input_shape=(1, X_train_tfidf_dense.shape[2])))
model_lstm.add(Dense(1, activation='sigmoid'))
```

- Détection d'anomalie avec Isolation Forest.

5 État des Travaux

- **Difficultés Rencontrées** :
  - Problèmes techniques ou méthodologiques
    * Il faudrait tenter d'autres design patterns de modèles. Mais j'ai encore du mal à me repérer dans ce que je peux utiliser dans l'analyse des requêtes http_url.
    * Je ne connais pas bien les techniques non supervisées comme Isolation Forest.
    * Je n'ai que des connaissances de base en séries temporelles, je ne me sens pas capable de faire du deep learning dessus pour l'instant.
    * La comparaison des modèles entre eux est très *artisanale*, à base de notebook.
    * La Tokenization des requêtes pourrait être améliorée.
    * Comment utiliser les autres *features* disponibles ? Les adresses IP, les ports, les timestamps ?
    * Comment prévoir une solution simple et rapide pour les analystes du SOC ?
  - Solutions envisagées ou mises en place
    * Utiliser MLFlow pour mieux comparer les modèles entre eux, et choisir le meilleur à mettre à disposition.
    * Si possible: Utiliser Spark en mode Streaming sur les données du SIEM. Ou alors, utiliser un programme effectuant un appel à l'API du SIEM. ?
    * Analyser d'autres articles sur le sujet (voir bibliographie)
- **Travaux Restants** :
  - Étapes à venir
    * Implémenter MLFlow
    * Tester différents algorithmes de ML et DL au sein de MLFlow.
    * Construire un outil permettant d'analyser une requête en particulier ou un lot de requêtes (Un exemple est proposé dans un des tutoriels de la plateforme Onyxia)
    * Documenter pour assurer une reproductibilité du projet.

  - Objectifs pour les prochaines semaines
    * Clarifier les modèles possibles, les entraîner avec recherche d'hyperparamètre sur MLFlow

6 Bibliographie

Le SOC étant une nouvelle unité de l'Insee, il n'y a pas eu de travaux sur le sujet en amont en son sein. J'ai axé ma recherche bibliographique autour des techniques de machine learning pour la cybersécurité, et autour des techniques de Deep Learning.

- **Références Utilisées** :
  - Articles scientifiques:
    * Isolation Forest, Liu, Ting, Zhou
    * R. Fontugne, P. Borgnat, P. Abry, K. Fukuda. "MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking". ACM CoNEXT 2010. Philadelphia, PA. December 2010.
  - Livres et manuels:
    * An introduction to statistical learning with applications in Python
    * Deep Learning Patterns and Practices, Andrew Ferlitsch
    * Deep Learning from Scratch - Building with Python from first principles, Seth Weidman
    * Deep Learning_ A Visual Approach – Andrew Glassner – Illustrated, 2021
    * Machine learning with Python cookbook, Kyle Gallatin & Chris Albon
    * Deep Learning, Ian Goodfellow
    * Machine Learning for cybersecurity, Emmanuel Tsukerman, (Chapter 6 Automatic Intrusion Detection)
    * Machine Learning for computer and Cyber Security, Principles, Algorithms, and Practices, Brij B. Gupta
  - Documentation technique
- **Ressources Additionnelles** :
  - Sites web
    * Scikit-Learn
  - Outils et bibliothèques spécifiques
    * Splunk
- Articles:
  - Isolation Forest, Liu, Ting, Zhou

  - R. Fontugne, P. Borgnat, P. Abry, K. Fukuda. “MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking”. ACM CoNEXT 2010. Philadelphia, PA. December 2010.
  - Levi Junior, David Macedo, Adriano Lorena Inácio de Oliveira, Cleber Zanchettin, “Detecting Malicious HTTP Requests Without Log Parser Using RequestBERTBiLSTM”, 2022

7 Conclusion

- Résumé des principales réalisations:
  - premières analyses de logs réalisées
  - plusieurs modèles testés
- Le stage se déroule bien. J’aimerais avoir un peu plus d’appui technique dans le choix des modèles.