Détection de Comportements Malveillants d’applications Windows

Contexte et scénario

Bienvenue dans cette épreuve « Data Challenge » d’analyse et de prédiction de comportements malveillants. Vous êtes invités à participer à un défi organisé par **l’Université PARIS I Panthéon-Sorbonne** et le **ComCyber du Ministère de l’Intérieur**, ce service est chargé de la lutte contre les cybermenaces, notamment au travers d’un de ses centres (CNENUM) qui parmi ses multiples missions, est en charge de l’analyse des systèmes et des logiciels malveillants dans le cadre d’incidents d’atteinte aux STAD (systèmes de traitement automatisés de données).

Et dans le cadre de ses travaux, le CNENUM a pu composer un large corpus de programmes Windows (des binaires au format PE ou DLL), dont certains sont bénins et d’autres, clairement malveillants. L’unité est donc confrontée à un besoin de classifier ces applications selon différents comportements potentiellement dangereux. L’enjeu est de taille : fournir aux analystes une solution rapide et fiable pour détecter des comportements suspects, voire malveillants, afin d’accélérer le processus d’analyse.

Description du jeu de données

Nous mettons à votre disposition un ensemble d’environ 20 000 binaires Windows (PE ou DLL).

Pour chacun de ces binaires, nous avons extrait les CFG (Control Flow Graphs), et les avons stockés dans des fichiers (un fichier par binaire).

Un CFG (Control Flow Graph) est une représentation en graphe qui montre tous les chemins possibles qu'un programme peut suivre lors de son exécution. Les nœuds représentent des blocs d’instructions, et les flèches indiquent les sauts vers des adresses d’instruction (comme des appels de fonction ou encore des boucles).

Chaque fichier représente, au format digraph, la suite d’instructions Intel x86 32bits ou 64bits présentes dans le binaire, ainsi que la structure des sauts (jumps), appels (calls) et retours (ret).

Exemple d’extrait (format digraph) :

```
Digraph G {
"10001000" [label = "10001000 : INST : mov eax, dword ptr [0x10003110]"]
"10001000" -> "10001005"
```

"10001005" [label = "10001005 : INST : mov dword ptr [0x10006000], eax"]

"10001005" -> "1000100a"

"1000100a" [label = "1000100a : RET : ret "]

...

}

Les instructions, comme mov eax, dword ptr [0x10003110]"] sont précédées par l'adresse contenant l'instruction (ici 10001000), cette instruction simple effectue l'opération suivante :

> Copie la valeur 4 octets (un « double mot » ou *dword*) qui se trouve à l'adresse mémoire 0x10003110 dans le registre EAX. Autrement dit, elle lit un entier 32 bits stocké en mémoire à cette adresse et le place dans EAX.

Les instructions x86 sont les commandes simples que le processeur exécute pour faire fonctionner un programme, comme déplacer des données, effectuer des calculs, ou gérer des conditions et des sauts. Elles sont écrites dans un langage proche du matériel appelé assembleur.

Quand un programme est compilé, son code source est transformé en binaire, une suite de 0 et 1 que le processeur comprend. Pour lire ces instructions, on utilise un outil appelé désassembleur (comme Ghidra, IDA Pro ou objdump). Ce logiciel traduit le binaire en instructions x86 lisibles.

Ces instructions sont utiles pour :

- Comprendre le fonctionnement d'un programme sans le code source.
- Déboguer ou analyser un logiciel.
- Étudier la sécurité ou analyser des logiciels malveillants.

Ces suites d'instructions représentant le CFG vous permettront de reconstruire la logique d'exécution du binaire de multiples manières (graphes, suites d'instructions, représentations sous forme d'images, etc.), vous avez donc le choix d'extraire la sémantique des programmes et d'exploiter leur structure au travers de leur graphe, ou encore de simplement exploiter la syntaxe en ne travaillant par exemple que sur les instructions, votre seule limite sera votre imagination.

En complément des fichiers Digraph de caractéristiques (features) contenant les instructions des programmes binaires**,** il vous sera remis un fichier d'étiquettes (label) contenant la liste des comportements observées. Ce fichier récapitulatif (CSV) contient des étiquettes au format « One-Hot encoding » liées aux comportements observés, ces comportements ont été observés par des sanbox d'analyse dynamique, cela signifie que chacun de ces binaires ont été exécutés dans un environnement d'exécution virtuel surveillé par des « sondes » afin de déterminer le comportement du programme à l'exécution, et ont attribué une étiquette comportementale au programme. Chaque ligne décrit un binaire identifié par son **hash** et présente diverses colonnes indiquant la présence (1) ou l'absence (0) d'un comportement. Donc une ligne par binaire, et une colonne par comportement. **A noter que les colonnes allant de B à D sont des métadonnées (num_antivirus_malicious ; first_submission_date ; suggested_threat_label) et qu'ils ne sont pas à prédire**.

Précisions :

- Environ la moitié de ces binaires sont considérés comme bénins, l'autre moitié étant malveillante. La colonne B représente le nombre d'antivirus d'antivirus ayant signalé comme malveillant le binaire.
- Les colonnes de comportements représentent des actions parfois suspectes (ex. : accès à la mémoire, communication réseau, modification de la base de registres, etc.).

Objectif du challenge

Vous devez mettre en place un **système de prédiction** (modèle d'IA ou autre technique) capable de déterminer pour chaque binaire **quels comportements** (parmi ceux décrits dans le fichier d'étiquettes) sont susceptibles d'être observés. Pour y parvenir, vous pourrez :

1. **Choisir** votre méthode de représentation des données (texte, graphes, images, séquences, etc...).
2. **Extraire** des **caractéristiques pertinentes** (features) depuis les fichiers digraph.
3. **Entraîner** un ou plusieurs modèles pour **prédire** les étiquettes comportementales.
4. **Évaluer** votre solution sur l'ensemble de test et remplissez le fichier csv de prédiction de l'ensemble de test. Nous évaluerons vos résultats avec la métrique de Macro F1.

Afin d'évaluer votre approche, un petit jeu de test vous sera fourni. Vous devrez réaliser les prédictions sur ce jeu et nous remettre le fichier CSV (qui vous sera fourni) contenant vos résultats au format One-Hot Encoding. Les performances obtenues à partir de ce fichier de prédictions seront utilisées pour comparer les solutions de chaque participant.

Consignes et livrables attendus

1. **Introduction / Documentation**
    - Décrivez votre approche, vos hypothèses de départ, la façon dont vous gérez les fichiers digraph (comment vous extrayez les informations utiles, etc.).
2. **Implémentation**
    - Fournissez votre **pipeline de traitement** (lecture des fichiers de feature, extraction de features, entraînement du modèle, prédiction).
3. **Résultats**
    - Fournissez le fichier csv complété pour la prédiction de l'ensemble de test.
4. **Remise du projet**
    - Tout code source pertinent (Script Python, répertoire Git, notebook Jupyter, etc.).

Critères d'évaluation

1. **Qualité** de l'extraction et de la transformation des données (capacité à exploiter correctement les fichiers au format digraph).

2. **Pertinence** de la méthode de modélisation (modèle d'IA, algorithme d'analyse).
3. **Performance** du modèle (métriques de classification).
4. **Absence de biais.**

Conclusion

Cette épreuve vous place dans la peau d'un **analyste data** / **IA** qui collabore avec les équipes forensiques de la Gendarmerie Nationale. Votre mission : concevoir un **système intelligent** capable de prédire des comportements malveillants à partir de **CFG** et d'instructions assembleur. Nous attendons **votre ingéniosité** et **votre créativité** pour imaginer des solutions novatrices, tout en gardant en tête la **fiabilité** indispensables pour lutter contre les menaces informatiques.

Nous vous souhaitons bonne chance dans cette aventure !
À vous de jouer et que le meilleur modèle gagne !