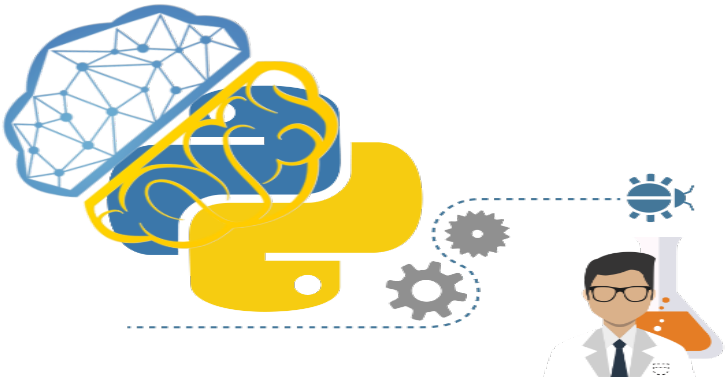
Université de Valenciennes et du Hainaut-Cambrésis

M2C - Malware Clustering et Classification



Vincent Romé - Axel Foulon - Julien Dupagny -

vrome/afoulon/jdupany@etu.univ-valenciennes.fr

— 9 Juin 2018 —

HISTORIQUE DES VERSIONS			
DATE	VERSION	ÉVOLUTION DU DOCUMENT	RÉDACTEUR
9/06/2017	0.1	Version préliminaire	Équipe complète

Table des matières

Notice légal

Le présent document présente et exprime, sauf indication contraire le fruit de la recherche de la réflexion, des idées, et du développement d'une solution pouvant répondre aux besoins et aux spécification du marché de la cyber sécurité en matière de détection de malware. Ce document doit être considéré dès lors comme les points de vue et interprétation des auteurs. Ce document ne reflète pas nécessairement l'état de la technologie la plus récente et pourrait faire l'objet de mises à jours.

Les sources tierces seront citées, le cas échéant, l'équipe du projet acceptera d'étudier toute requête en cas d'oubli cependant elle ne reste pas responsable du contenu des sources extérieures. Le dit document a une vocation strictement informative. Tout personne possédant un exemplaire de ce document pourra être tenu responsable de l'usage qu'il pourrait en faire.

Introduction

Jusqu'à présent les systèmes de détection d'intrusion reposaient traditionnellement sur des signatures générées manuellement par des experts en sécurité, puis nous avons vu apparaître des systèmes permettant de détecter des patterns entre des jeux de données ceci à permis de générer automatiquement ces règles. Aujourd'hui avec l'apparition du "Big Data" des solutions comme le Deep Learning ou Machine learning sont souvent présentés comme les technologies pouvant révolutionner les systèmes de détections et les performances. Ces systèmes permettent de générer automatique un modèle de détection à partir de données et leurs capacité à généraliser les événements malveillants permettait en effet de détecter des éléments encore inconnus. L'objectif est d'ici comprendre le fonctionnement de ces algorithmes et de les appliquer au milieu de la sécurité informatique. D'exposer les résultats de notre recherche sur la détection de fichier PE (exécutable windows malveillants). Tout en gardant un regard critique sur les résultats et en essayant d'apporter des solution sur l'utilisation de tel algorithme en production.

Un modèle de détection supervisé est construit à partir de données labellisées fournies par l'expert : des événements bénins, mais aussi des événements malveillants pour guider le modèle de détection. L'algorithme d'apprentissage va automatiquement chercher les points permettant de caractériser chacune des classes ou de les discriminer pour construire le modèle de détection

Abstract

Contexte du projet

Les contraintes

Il est dès à présent de prendre en compte plusieurs contraintes que nous ne ne devons pas perdre de vu :

- Prédiction rapide
- Mise à jour périodique du modèle
- Interprétable (Expert puisse comprendre le modèle et l'ajuster)

Les objectifs

A travers ce projet nous voulons fournir un solution permettant la classification et une détection via des alorithmes de machine learning de malwares touchant les systèmes d'exploitation Windows de Microsoft.

Nous pourrons découvrir le principe de base d'une analyse statique de fichier malveillant, ainsi que les notions relative au machine learning et la data science pour le traitement de gros flux de données. Il s'agit d'une application particulière en matière de sécurité informatique des nouveaux algorithmes qui ont le vent en poupe.

L'objectif est qu'a l'issue de ce projet nous soyont en mesure de comprendre les notions de base relative à :

- L'analyse de malware
- Les algorithmes d'apprentissage automatique
- Comprendre le workflow associé à l'utilisation de tel algorithme.

A l'issue de ce projet un PoC d'implémentation en python devra être disponible. Des premiers résultats permettant de conclure sur l'efficacité et la complexité ainsi que les limitions de tel algorithmes appliqué dans le domaine de la sécurité.

Implémentation

7.1 Format des données

Afin de construire notre modèle de détection via du machine learning il est nécessaire de récolter des données d'apprentissage. il faut donc construire un dataset.

Pour constituer notre dataset servira par la suite à l'entraintement de notre algorithme de machine learning, nous allons essayer de générer une collection de fichier au format JSON. Dans ces fichiers chaque ligne contiendra un objet JSON. Nous avons tenté de définir une structure permettant de regrouper suffisamment d'informations.

Chacun des ces objets inclus les types de données suivant :

Il y a quelques grands principes à respecter lors de la construction du jeu d'apprentissage. Tout d'abord, il doit comporter un nombre suffisant d'instances pour que le modèle de détection soit capable de généraliser correctement les comportements bénin et malveillant.

Exemple de fichier JSON généré pour un fichier :


```
{
        "size": 106496,
        "path": "/home/light/Documents/Cours_CDSI/ML/dataset/theZoo/malware
        "name": "0008065861f5b09195e51add72dacd3c4bbce6444711320ad349c7dab5
        "appeared": "2017-01",
        "label": "-1",
        "hashes": {
                "md5": "d2074d6273f41c34e8ba370aa9af46ad",
                "sha1": "5074ec3ca672f74ea05a7b5f0f52339fbf440f9b",
                "sha256": "0008065861f5b09195e51add72dacd3c4bbce6444711320a
        },
        "nb_sections": 6,
        "nb_exported_fonctions": 7,
        "nb_imported_fonctions": 95,
        "exported_fonctions": ["CON", "Fdown", "Fdown2", "InetReadF", "_Com
        "imported_fonctions": ["InternetOpenA", "DeleteUrlCacheEntry", "Int
        "imported_libraries": ["WININET.dll", "KERNEL32.dll", "ADVAPI32.dll
        "general": {
                "vsize": 118784,
                "has_debug": 1,
                "exports": 7,
                "imports": 95,
                "has_relocations": 1,
```

```
        "has_resources": 1,
        "has_signature": 0,
        "has_tls": 0,
        "symbols": 0,
        "entrypoint": "0x100055dc"
    },
    "header": {
        "coff": {
            "timestamp": 1383637370,
            "machine": "I386",
            "characteristics": ["CHARA_32BIT_MACHINE", "DLL",
        },
        "optional": {
            "subsystem": "WINDOWS_GUI",
            "dll_characteristics": [],
            "magic": "PE32",
            "major_image_version": 0,
            "minor_image_version": 0,
            "major_linker_version": 7,
            "minor_linker_version": 10,
            "major_operating_system_version": 4,
            "minor_operating_system_version": 0,
            "major_subsystem_version": 4,
            "minor_subsystem_version": 0,
            "sizeof_code": 65536,
            "sizeof_headers": 4096,
            "sizeof_heap_commit": 4096
        }
    },
    "sections": [{
        "name": ".text",
        "characteristics": 1610612768,
        "vsize": "0xf2eb",
        "size": "0x10000",
        "vaddres": "0x1000",
        "entropy": 6.625413677157515
    }, {
        "name": ".rdata",
        "characteristics": 1073741888,
        "vsize": "0x3a3b",
        "size": "0x4000",
        "vaddres": "0x11000",
        "entropy": 5.043471612579925
    }, {
        "name": ".data",
        "characteristics": 3221225536,
        "vsize": "0x3358",
        "size": "0x1000",
        "vaddres": "0x15000",
        "entropy": 2.7813728969479183
```

```
        }, {
                "name": ".SHARDAT",
                "characteristics": 3489660992,
                "vsize": "0x8",
                "size": "0x1000",
                "vaddres": "0x19000",
                "entropy": -0.0
        }, {
                "name": ".rsrc",
                "characteristics": 1073741888,
                "vsize": "0x368",
                "size": "0x1000",
                "vaddres": "0x1a000",
                "entropy": 3.2077393530680016
        }, {
                "name": ".reloc",
                "characteristics": 1107296320,
                "vsize": "0x1630",
                "size": "0x2000",
                "vaddres": "0x1b000",
                "entropy": 5.857932346902008
        }]
}
```

Le Format PE

Nous avons décidé de travailler uniquement sur des malwares Windows afin de ne pas compliquer le sujet. Nous sommes partie du constat que la majorité des personnes utilisent ce système d'exploitation. Le format PE (Portable Executable, executable portable) est le format predominant des fichiers exécutables et des bibliothèques sur les systèmes d'exploitation Windows 32 bits et 64 bits. Ce format est utilisé chez Microsoft pour les pilotes, les programmes mais aussi les DLL et autres fichiers exécutables. Ce format est dit portable car il peut être porté sous les différents systèmes que Windows NT supporte et il est cross architecture, Il supporte des architectures (ARM, AMD et Intel).

Le Fuzzy hashing

Il s'agit d'un outil dit de fuzzy hashing ceci signifie qu'une valeur de hash qui essaye de détecter le niveau de similarité entre deux fichier au niveau binaire. Ce type de hash est différent d'un hash cryptographique tel que SHA1. Un hash cryptographique standard permet de répondre à la question " c'est deux fichiers sont-ils identique ?" Un fuzzy hash aussi appelé similarity hash lui est utile pour répondre à la question "une partie de ce fichier est-elle là même que ce second ?" Les deux grands algorithme de fuzzy hashing utilisé pour la classification de malware sont :

- **ssdeep :**
- **machocke :** Il s'agit d'un algorithme de fuzzy hashing basé sur le CFG

il peut donc être intéressant d'utiliser ce hash comme feature. Nous n'avons malheureusement pas eu le temps de tester cette possibilité.

Conclusion

À travers cet article nous pouvons en conclure : Le machine learning c'est pas magique, un gros travaille de featuring est nécessaire. Un bon jeu de donnée de départ est aussi très important. Le multi-architecture et l'extension vers d'autre type de virus que ceux ciblant Windows est facilement envisageable grâce à la library LIEF. Le machine learning est très utile pour faire un premier filtre afin de catégoriser un gros jeu de données comparé à des algorithmes de fuzzy hashing. L'utilisation de LightGBM L'état actuel de nos travaux nous permet de conclure que cette méthode est pertinente et efficace car les résultats obtenus sont déjà très bons alors que de nombreuses pistes sont encore disponibles pour l'améliorer. Nous avons implémenté uniquement une solution permettant de faire du clustering nous aimerions nous tourner vers des algorithmes de classification.

Définition

Machine Learning : retrouver des patterns dans un jeu de donnée afin de les regrouper.

Classification : Classer les données dans des catégories prédéfinies. Les algorithmes sont par exemple : decision Tree, Random forest…

Clustering : Regrouper les données dans un ensemble de catégories. Les algorithmes sont par exemple : KMEans HCA DBSCAN…

Feature : Caractéristique d'un objet utile pour l'algorithme (les patterns potentiels)

Vector of features : Un tableau de features.

Cluster : Un groupe d'objets décider par l'algorithme

Label : Nom du cluster.

CFG : En informatique, un graphe de flot de contrôle (abrégé en GFC, control flow graph ou CFG en anglais) est une représentation sous forme de graphe de tous les chemins qui peuvent être suivis par un programme durant son exécution.

JSON : JavaScript Object Notation (JSON) est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML par exemple