BÁO CÁO LAB

Môn học: Phương pháp học máy trong an toàn thông tin

Tên chủ đề: Lab 3

GVHD: Nguyễn Hữu Quyền

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT522.O21.ATCL.1

STT	Họ và tên	MSSV	Email
1	Nguyễn Đại Nghĩa	21521182	21521182@gm.uit.edu.vn
2	Hoàng Gia Bảo	21521848	21521848@gm.uit.edu.vn
3	Trương Đặng Văn Linh	21520328	21520328@gm.uit.edu.vn
4	Mai Quốc Cường	21521901	21521901@gm.uit.edu.vn

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

BÁO CÁO CHI TIẾT

Chuẩn bị dữ liệu:

```
!wget https://didierstevens.com/files/software/pdfid_v0_2_8.zip
```

```
--2024-05-12 14:06:28--  https://didierstevens.com/files/software/pdfid_v0_2_8.zip
Resolving didierstevens.com (didierstevens.com)... 96.126.103.196
Connecting to didierstevens.com (didierstevens.com)|96.126.103.196|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11844 (12K) [application/zip]
Saving to: ‘pdfid_v0_2_8.zip’

pdfid_v0_2_8.zip    100%[===================>]  11.57K  --.-KB/s    in 0s

2024-05-12 14:06:28 (141 MB/s) - ‘pdfid_v0_2_8.zip’ saved [11844/11844]
```

```
[ ] from zipfile import ZipFile
    # Donwload dataset.zip
    !gdown 1rMM1NEgZPhZFvZU8Iugxn6yCCvsIn-CI
    # Download data.zip
    !gdown 1cDRQ50SKHN6HtbwfxD8MNsqtZ2uvL-ai
    # Download Benign PE Samples Amber.7z
    # !gdown 1G5aIFGFbse9LCDqz9vI7eqKfrS1ZdBTC

    with ZipFile('data.zip', 'r') as zip_file:
      zip_file.extractall()
    with ZipFile('dataset.zip', 'r') as zip_file:
      zip_file.extractall()
    with ZipFile('pdfid_v0_2_8.zip', 'r') as zip_file:
      zip_file.extractall('pdfid')


    # with ZipFile('/content/drive/MyDrive/NT522/Lab3/data.zip', 'r') as zip_file:
      # zip_file.extractall()
    # with ZipFile('/content/drive/MyDrive/NT522/Lab3/dataset.zip', 'r') as zip_file:
    #   zip_file.extractall()
    # with ZipFile('pdfid_v0_2_8.zip', 'r') as zip_file:
    #   zip_file.extractall('pdfid')
```

```
Downloading...
From: https://drive.google.com/uc?id=1rMM1NEgZPhZFvZU8Iugxn6yCCvsIn-CI
To: /content/dataset.zip
100% 10.9M/10.9M [00:00<00:00, 50.2MB/s]
Downloading...
From (original): https://drive.google.com/uc?id=1cDRQ50SKHN6HtbwfxD8MNsqtZ2uvL-ai
From (redirected): https://drive.google.com/uc?id=1cDRQ50SKHN6HtbwfxD8MNsqtZ2uvL-ai&confirm=t&uuid=99058357-b5cb-4055-a67c-7a8c818a562f
To: /content/data.zip
100% 227M/227M [00:01<00:00, 124MB/s]
```

```
!pip install -q py7zr
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.6/67.6 kB 2.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 38.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 411.2/411.2 kB 27.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 138.9/138.9 kB 14.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.7/49.7 kB 5.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 93.1/93.1 kB 9.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.0/3.0 MB 55.1 MB/s eta 0:00:00
```

```
[ ] !pip install -q nltk
```

```python
import os
import shutil
from py7zr import SevenZipFile

folder_path = "/content"

for filename in os.listdir(folder_path):
  if filename.endswith(".7z"):
    file_path = os.path.join(folder_path, filename)

    if "DA Logs Benign" in file_path:
      output_folder = 'DA Logs Benign'
      os.makedirs(output_folder, exist_ok=True)

      with SevenZipFile(file_path, mode="r") as z:
        z.extractall(output_folder)

      for root, dirs, files in os.walk(output_folder):
        for f in files:
            src = os.path.join(root, f)
            dst = os.path.join(os.path.join(folder_path, output_folder), f)
            shutil.move(src, dst)

      objects = os.listdir(output_folder)
      for obj in objects:
        if os.path.isdir(output_folder + '/'+obj):
          shutil.rmtree(output_folder + '/'+obj)
    elif "DA Logs Malware" in file_path:
      output_folder = 'DA Logs Malware'
      os.makedirs(output_folder, exist_ok=True)
      with SevenZipFile(file_path, mode="r") as z:
        z.extractall(output_folder)
      for root, dirs, files in os.walk(output_folder):
        for f in files:
            src = os.path.join(root, f)
            dst = os.path.join(os.path.join(folder_path, output_folder), f)
            shutil.move(src, dst)
```

```
        objects = os.listdir(output_folder)
        for obj in objects:
          if os.path.isdir(output_folder + '/'+obj):
            shutil.rmtree(output_folder + '/'+obj)
      elif "JavascriptSamplesNotObfuscated" in file_path:
        # output_folder = 'JavascriptSamples'
        # os.makedirs(output_folder, exist_ok=True)
        with SevenZipFile(file_path, mode='r') as z:
          z.extractall()
      elif "JavascriptSamplesObfuscated" in file_path:
        output_folder = 'ObfuscatedJavascriptSamples'
        # os.makedirs(output_folder, exist_ok=True)
        with SevenZipFile(file_path, mode='r') as z:
          z.extractall()
        os.rename('JavascriptSamplesObfuscated', output_folder)
      else:
        with SevenZipFile(file_path, 'r') as zip_file:
          zip_file.extractall()
```

Câu 1: Cho biết kết quả accuracy và confusion matrix.

Định nghĩa đường dẫn 2 thư mục Javascripts đã giải nén:

```
[ ] import os
    from sklearn.feature_extraction.text import HashingVectorizer, TfidfTransformer
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score, confusion_matrix
    from sklearn.pipeline import Pipeline
```

```
js_path = "JavascriptSamples"
obfuscated_js_path = "ObfuscatedJavascriptSamples"
corpus = []
labels = []

file_types_and_labels = [(js_path, 0), (obfuscated_js_path, 1)]
```

Gán nhãn cho chúng:

```
[ ]  for files_path, label in file_types_and_labels:
         files = os.listdir(files_path)
         for file in files:
             file_path = files_path + "/" + file
             try:
                 with open(file_path, "r") as myfile:
                     data = myfile.read().replace("\n", "")
                     data = str(data)
                     corpus.append(data)
                     labels. append(label)
             except:
                 pass
```

Chia tập dữ liệu thành tập huấn luyện và tập thử nghiệm, đồng thời tạo pipeline cho NLP, tiếp theo sử dụng phân loại random forest:

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(
     corpus, labels, test_size=0.33, random_state=42
     )
     text_clf = Pipeline(
     [
     ("vect", HashingVectorizer(input="content", ngram_range=(1,
     3))),
     ("tfidf", TfidfTransformer(use_idf=True,)),
     ("rf", RandomForestClassifier(class_weight="balanced")),
     ]
     )
```

Chạy huấn luyện và cho ra đánh giá:

```
text_clf.fit(X_train, y_train)
y_test_pred = text_clf.predict(X_test)

print(accuracy_score(y_test, y_test_pred))
print(confusion_matrix(y_test, y_test_pred))
```

```
0.9587073608617595
[[604  31]
 [ 15 464]]
```

Câu 2: Cho biết kết quả vector X (Trích xuất thuộc tính tập tin PDF)

Import IPython để thu thập các output của script:

```
from IPython.utils import io
```

Định nghĩa hàm trích xuất thuộc tính. Chạy pdfid đọc một tập và lấy kết quả output của chúng. Kế tiếp, phân tích output để lấy vector số:

```
def PDF_to_FV(file_path):
    """Featuize a PDF file using pdfid."""
    with io.capture_output() as captured:
        %run -i pdfid/pdfid.py $file_path
    out = captured.stdout
    out1 = out.split("\n")[2:-2]
    return [int(x.split()[-1]) for x in out1]
```

Import listdir để liệt kê các tập tin của thư mục PDF và cho vào vòng lặp để trích xuất, quét hết tất cả tập tin vào mảng X:

```
X = []
PDFs_path = "PDFSamples"
files = os.listdir(PDFs_path)
for file in files:
    file_path = PDFs_path + '/'  + file
    X.append(PDF_to_FV(file_path))

print(X)
```

```
[[1096, 1095, 1061, 1061, 0, 0, 2, 32, 0, 43, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0], [153, 153, 82, 82, 2, 2, 2, 7, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0]]
```

Câu 3: Cho biết kết quả vector X (Trích xuất N-grams bằng cách sử dụng thuật toán hash-gram)

Chỉ định thư mục cần trích xuất, tham số N, import thư viện để hash và trích xuất N-grams từ chuỗi:

```
from os import listdir
from nltk import ngrams
import hashlib
directories = ["Benign PE Samples", "Malicious PE Samples"]
N = 2
```

Tạo các hàm đọc tập tin và chuyển chúng vào N-grams:

```
def read_file(file_path):
    """Reads in the binary sequence of a binary file."""
    with open(file_path, "rb") as binary_file:
        data = binary_file.read()
    return data

def byte_sequence_to_Ngrams(byte_sequence, N):
    return ngrams(byte_sequence, N)
```

Tiến hành hash N-grams:

```
def hash_input(inp):
    """Compute the MD5 hash of an input."""
    return int(hashlib.md5(inp).hexdigest(), 16)

def make_ngram_hashable(Ngram):
    """Convert N-gram into bytes to be hashable"""
    return bytes(Ngram)
```

Hàm hash_file_Ngrams_into_dictionary lấy một N-grams, hash nó, sau đó tăng số lượng count trong dict cho hàm băm. Module B đảm bảo không thể có nhiều hơn B khoá trong dict:

```
[ ] def hash_file_Ngrams_into_dictionary(file_Ngrams, T):
        """Hashes N-grams in a list and then keep track of the count in a dictionary"""
        for Ngram in file_Ngrams:
            hashable_Ngram = make_ngram_hashable(Ngram)
            hashed_and_reduced = hash_input(hashable_Ngram) % B
            T[hashed_and_reduced] = T.get(hashed_and_reduced, 0) + 1
```

Giá trị B là số nguyên tố lớn nhất nhỏ hơn 2^16 và tạo dict rỗng. Tiếp theo lặp lại qua các tập tin để count N-grams đã hash:

```
B = 65521
T = {}
for dataset_path in directories:
    samples = [f for f in listdir(dataset_path)]
    for file in samples:
        file_path = dataset_path + "/" + file
        file_byte_sequence = read_file(file_path)
        file_Ngrams = byte_sequence_to_Ngrams(file_byte_sequence, N)
        hash_file_Ngrams_into_dictionary(file_Ngrams, T)
```

Chọn 1000 N-gram phổ biến sử dụng với heapq:

```
K1 = 1000
import heapq
K1_most_common_Ngrams_Using_Hash_Grams = heapq.nlargest(K1, T)
```

Sau khi chọn top N-grams, được băm, tạo bộ thuộc tính N-grams,làm tăng vector đặc trưng:

```
def featurize_sample(file, K1_most_common_Ngrams_Using_Hash_Grams):
    """Takes a sample and produces a feature vector.
    The features are the counts of the K1 N-grams we've selectd"""
    K1 = len(K1_most_common_Ngrams_Using_Hash_Grams)
    fv = K1 * [0]
    file_byte_sequence = read_file(file_path)
    file_Ngrams = byte_sequence_to_Ngrams(file_byte_sequence, N)
    for Ngram in file_Ngrams:
        hashable_Ngram = make_ngram_hashable(Ngram)
        hashed_and_reduced = hash_input(hashable_Ngram) % B
        if hashed_and_reduced in K1_most_common_Ngrams_Using_Hash_Grams:
            index = K1_most_common_Ngrams_Using_Hash_Grams.index(hashed_and_reduced)
            fv[index] += 1
    return fv
```

Tạo bộ dữ liệu:

```
X = []
for dataset_path in directories:
    samples = [f for f in listdir(dataset_path)]
for file in samples:
    file_path = dataset_path + "/" + file
    X.append(featurize_sample(file_path, K1_most_common_Ngrams_Using_Hash_Grams))
```

```
# len(X[0])
X[0]
```

Kết quả nhận được:

```
[3,
 18,
 0,
 21,
 1,
 1,
 47,
 2,
 4,
 3,
 2,
 11,
 0,
 8,
 6,
 2,
 1,
 0,
 4,
 1,
 15,
 236,
 23,
 19,
 1,
 1,
 0,
 73,
 13,
 14,
 2,
 2,
 8,
 4,
 40,
 0,
 4,
 8,
```

Câu 4: Cho biết kết quả đánh giá (Xây dựng bộ phân loại động phần mềm độc hại)

Do tập dữ liệu là các tập tin nhật ký định dạng JSON nên import thư viện JSON:

```
[ ]  import numpy as np
     import os
     import json

     directories_with_labels = [("DA Logs Benign", 0), ("DA Logs Malware", 1)]
```

Viết hàm parse nhật ký JSON:

```
def get_API_class_method_type_from_log(log):
    """Parse out API calls from behavioral logs."""
    API_data_sequence = []
    with open(log) as log_file:
        json_log = json.load(log_file)
        api_calls_array = "[" + json_log["api_calls"] + "]"
        # 3. Chọn trích xuất class, method và type từ API call
        api_calls = json.loads(api_calls_array)
        for api_call in api_calls:
            data = api_call["class"] + ":" + api_call["method"] + ":" + api_call["type"]
            API_data_sequence.append(data)
        return API_data_sequence
```

Sau đó phân nhãn:

```
data_corpus = []
labels = []
for directory, label in directories_with_labels:
    logs = os.listdir(directory)
    for log_path in logs:
        file_path = directory + "/" + log_path
        try:
            data_corpus.append(get_API_class_method_type_from_log(file_path))
            labels.append(label)
        except:
            pass
print(data_corpus[0])
```

```
['javax.crypto.Cipher:doFinal:crypto', 'android.app.SharedPreferencesImpl$EditorImpl:putInt:globals', 'android.app.SharedPreferencesImpl$EditorImpl:putLong:globals', 'libcore.io.IoBridge:open:file', 'libcore.io.IoBridge:open:file', '
```

Chia tập dữ liệu huấn luyện và kiểm thử:

```
[ ]  from sklearn.model_selection import train_test_split
     corpus_train, corpus_test, y_train, y_test = train_test_split(
         data_corpus, labels, test_size=0.2, random_state=11
     )
```

Trích xuất N-grams:

```
import collections
from nltk import ngrams
import numpy as np
def read_file(file_path):
    """Reads in the binary sequence of a binary file."""
    with open(file_path, "rb") as binary_file:
        data = binary_file.read()
    return data
def text_to_Ngrams(text, n):
    """Produces a list of N-grams from a text."""
    Ngrams = ngrams(text, n)
    return list(Ngrams)
def get_Ngram_counts(text, N):
    """Get a frequency count of N-grams in a text."""
    Ngrams = text_to_Ngrams(text, N)
    return collections.Counter(Ngrams)
```

Xác định N=4 và thua thập tất cả N-grams:

```
N = 4
total_Ngram_count = collections.Counter([])
for file in corpus_train:
    total_Ngram_count += get_Ngram_counts(file, N)
```

Sau đó thu hẹp K1=3000 và N-grams phổ biến:

```
[ ] K1 = 3000
    K1_most_frequent_Ngrams = total_Ngram_count.most_common(K1)
    K1_most_frequent_Ngrams_list = [x[0] for x in K1_most_frequent_Ngrams]
```

Viết một hàm phân loại mội mẫu thhanfh một vector có N-grams:

```
def featurize_sample(file, Ngrams_list):
    """Takes a sample and produces a feature vector.
    The features are the counts of the K1 N-grams we've selected.
    """
    K1 = len(Ngrams_list)
    feature_vector = K1 * [0]
    fileNgrams = get_Ngram_counts(file, N)
    for i in range(K1):
        feature_vector[i] = fileNgrams[Ngrams_list[i]]
    return feature_vector
```

Sử dụng hàm trên để tạo bộ huấn luyện và mẫu kiểm thử:

```
X_train = []
for sample in corpus_train:
    X_train.append(featurize_sample(sample,
K1_most_frequent_Ngrams_list))
X_train = np.asarray(X_train)
X_test = []
for sample in corpus_test:
    X_test.append(featurize_sample(sample,
K1_most_frequent_Ngrams_list))
X_test = np.asarray(X_test)
```

Tiếp tục thu hẹp K1=3000 N-12 grams thành K2=500. Sau đó thiết lập pipeline chạy phân loại XGBoost:

```
[ ] from sklearn.feature_selection import SelectKBest, mutual_info_classif
    from sklearn.pipeline import Pipeline
    from xgboost import XGBClassifier
    K2 = 500
    mi_pipeline = Pipeline(
        [
            ("mutual_information", SelectKBest(mutual_info_classif, k=K2)),
            ("xgb", XGBClassifier()),
        ]
    )
```

Sau đó huấn luyện và kiểm thử tập train, test:

```
[ ]  mi_pipeline.fit(X_train, y_train)
     print("Training accuracy:")
     print(mi_pipeline.score(X_train, y_train))
     print("Testing accuracy:")
     print(mi_pipeline.score(X_test, y_test))
```

```
Training accuracy:
0.892513529765484
Testing accuracy:
0.8244137101623572
```

Câu 5: Cho biết kết quả đánh giá mô hình qua tập test (MalConv - Quy trình áp dụng sâu cho phát hiệ phần mềm độc hại PE)

Import thư viên numpy để tính toán vector và tqdm để theo dõi tiền trình trong vòng lặp:

```
[ ]  import numpy as np
     from tqdm import tqdm
```

Định nghĩa hàm để chuyển byte thành vector:

```
[ ]  def embed_bytes(byte):
         binary_string = "{0:08b}".format(byte)
         vec = np.zeros(8)
         for i in range(8):
             if binary_string[i] == "1":
                 vec[i] = float(1) / 16
             else:
                 vec[i] = -float(1) / 16
         return vec
```

Đọc các tin PE mẫu và dán nhãn cho chúng:

```
[ ]  import os
     from os import listdir
     directories_with_labels = [("Benign PE Samples", 0), ("Malicious PE Samples", 1)]
     list_of_samples = []
     labels = []
     for dataset_path, label in directories_with_labels:
         samples = [f for f in listdir(dataset_path)]
         for file in samples:
             file_path = os.path.join(dataset_path, file)
             list_of_samples.append(file_path)
             labels.append(label)
```

Định nghĩa hàm đọc chuỗi byte trong tập tin:

```
[ ]  def read_file(file_path):
         """Read the binary sequence of a file."""
         with open(file_path, "rb") as binary_file:
             return binary_file.read()
```

Đặt độ dài tối đa, maxSize byte, để đọc cho mỗi mẫu, lấy tất cả byte của mẫu đưa vào x:

```
[ ]  max_size = 15000
     num_samples = len(list_of_samples)
     X = np.zeros((num_samples, 8, max_size))
     Y = np.asarray(labels)
     file_num = 0
     for file in tqdm(list_of_samples):
         sample_byte_sequence = read_file(file)
         for i in range(min(max_size, len(sample_byte_sequence))):
             X[file_num, :, i] = embed_bytes(sample_byte_sequence[i])
         file_num += 1
```

```
100%|██████████| 87/87 [00:12<00:00,  7.20it/s]
```

Thiết lập tình tối ưu:

```
import tensorflow as tf

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.01,
    decay_steps=10000,
    decay_rate=0.9)

my_opt = tf.keras.optimizers.legacy.SGD(learning_rate=lr_schedule, decay=1e-5, nesterov=True)
```

Sử dụng API của keras để thiết lập màn thần kinh học sâu:

```
from keras import Input
inputs = Input(shape=(8, max_size)) # shape: specify the number `n` that the expected input will be batches of n-dimensional vectors.

from keras.layers import Conv1D
conv1 = Conv1D(kernel_size=(128), # Size of kernel, this will be the height for Filter
               filters=32, # number of filter
               strides=(128), # distance between 2 kernels when scanning. In other words, it modifies the amount of movement over the image or video
               padding="same" #  the amount of pixels added to an matrix to be suitable when this matrix being processed by the kernel of a CNN
               # 'same' means padding with zeros evenly to the left/right or up/down of the input
              )(inputs)
conv2 = Conv1D(kernel_size=(128),
               filters=32,
               strides=(128),
               padding="same"
              )(inputs)

from keras.layers import Activation
a = Activation("sigmoid", name="sigmoid")(conv2) # Use 'sigmoid' as activate function for this layer

from keras.layers import multiply
mul = multiply([conv1, a]) # Apply multiply operator (*) to 'conv1' and 'a'
b = Activation("relu", name="relu")(mul) # Use 'relu' as activate function for this layer

from keras.layers import GlobalMaxPool1D
p = GlobalMaxPool1D()(b) # Global 'max pooling' operation for 1D temporal data.

from keras.layers import Dense
d = Dense(16)(p) # Dense layer, number of neurons
predictions = Dense(1, activation="sigmoid")(d)

from keras import Model
model = Model(inputs=inputs, outputs=predictions)
```

Biên dịch mô hình và chọn batch size:

```
model.compile(optimizer=my_opt, loss="binary_crossentropy", metrics=["acc"]) # Train the model, use 'binary_crossentropy' as a loss function
model.summary()

batch_size = 16 # the number of data samples in one training session.
num_batches = int(num_samples / batch_size)
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 8, 15000)]        0         []

 conv1d_1 (Conv1D)           (None, 1, 32)             6144003   ['input_1[0][0]']
                                                       2

 conv1d (Conv1D)             (None, 1, 32)             6144003   ['input_1[0][0]']
                                                       2

 sigmoid (Activation)        (None, 1, 32)             0         ['conv1d_1[0][0]']

 multiply (Multiply)         (None, 1, 32)             0         ['conv1d[0][0]',
                                                                  'sigmoid[0][0]']

 relu (Activation)           (None, 1, 32)             0         ['multiply[0][0]']

 global_max_pooling1d (Glob  (None, 32)                0         ['relu[0][0]']
 alMaxPooling1D)

 dense (Dense)               (None, 16)                528       ['global_max_pooling1d[0][0]']

 dense_1 (Dense)             (None, 1)                 17        ['dense[0][0]']

==================================================================================================
Total params: 122880609 (468.75 MB)
Trainable params: 122880609 (468.75 MB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________________________
```

Huấn luyện mô hình:

```
# Evaluation
print(model.evaluate(X, Y))
```

```
3/3 [==============================] - 2s 326ms/step - loss: 0.6890 - acc: 0.8276
[0.6890001893043518, 0.8275862336158752]
```

Câu 6: Cài đặt. UPX và tiến hành đóng gói các tập tin pe tại Benign PE Samples UPX (Xử lý phần mềm độc hại packer)

Bước đầu là em cài đặt upx:

```
[ ] !apt-get install upx

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'upx-ucl' instead of 'upx'
The following additional packages will be installed:
  libucl1
The following NEW packages will be installed:
  libucl1 upx-ucl
0 upgraded, 2 newly installed, 0 to remove and 45 not upgraded.
Need to get 479 kB of archives.
After this operation, 2,176 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libucl1 amd64 1.03+repack-6 [25.8 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 upx-ucl amd64 3.96-3 [453 kB]
Fetched 479 kB in 3s (187 kB/s)
Selecting previously unselected package libucl1:amd64.
(Reading database ... 121918 files and directories currently installed.)
Preparing to unpack .../libucl1_1.03+repack-6_amd64.deb ...
Unpacking libucl1:amd64 (1.03+repack-6) ...
Selecting previously unselected package upx-ucl.
Preparing to unpack .../upx-ucl_3.96-3_amd64.deb ...
Unpacking upx-ucl (3.96-3) ...
Setting up libucl1:amd64 (1.03+repack-6) ...
Setting up upx-ucl (3.96-3) ...
update-alternatives: error: no alternatives for upx
update-alternatives: using /usr/bin/upx-ucl to provide /usr/bin/upx (upx) in auto mode
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link
```

Tiến hành chạy upx:

```python
import os
from subprocess import Popen, PIPE

files_path = "/content/drive/MyDrive/Benign PE Samples UPX/"
files = os.listdir(files_path)
file_paths = [files_path + x for x in files]  # Ensure proper path concatenation

cmd = "upx"

for path in file_paths:
    cmd2 = f'{cmd} "{path}"'
    process = Popen(cmd2, stdout=PIPE, stderr=PIPE, shell=True)
    output, errors = process.communicate()

    # Check for errors in output
    if "error" in output.decode() or errors:
        print(f"Error processing {path}: {errors.decode()}")
    else:
        print(f"Success: {output.decode()}")
```

Kết quả nhận được:

```
Error processing /content/drive/MyDrive/Benign PE Samples UPX/fhmanagew.exe: upx: /content/drive/MyDrive/Benign PE Samples UPX/fhmanagew.exe: AlreadyPackedException: already packed by UPX
Error processing /content/drive/MyDrive/Benign PE Samples UPX/finger.exe: upx: /content/drive/MyDrive/Benign PE Samples UPX/finger.exe: AlreadyPackedException: already packed by UPX
Error processing /content/drive/MyDrive/Benign PE Samples UPX/find.exe: upx: /content/drive/MyDrive/Benign PE Samples UPX/find.exe: AlreadyPackedException: already packed by UPX
Error processing /content/drive/MyDrive/Benign PE Samples UPX/fltMC.exe: upx: /content/drive/MyDrive/Benign PE Samples UPX/fltMC.exe: AlreadyPackedException: already packed by UPX
Error processing /content/drive/MyDrive/Benign PE Samples UPX/fixmapi.exe: upx: /content/drive/MyDrive/Benign PE Samples UPX/fixmapi.exe: AlreadyPackedException: already packed by UPX
Error processing /content/drive/MyDrive/Benign PE Samples UPX/FirstLogonAnim.exe: upx: /content/drive/MyDrive/Benign PE Samples UPX/FirstLogonAnim.exe: AlreadyPackedException: already packed b
Error processing /content/drive/MyDrive/Benign PE Samples UPX/fc.exe: upx: /content/drive/MyDrive/Benign PE Samples UPX/fc.exe: AlreadyPackedException: already packed by UPX
```

Thông quá thông báo lỗi có thể thấy được là các file đã được đóng gói sẵn bởi upx rồi nên không thể đóng gói thêm nữa.

Bây giờ em sẽ tiến hành giải nén file đã đóng gói với câu lệnh “upx -d”:

```
import os
from subprocess import Popen, PIPE

files_path = "/content/drive/MyDrive/Benign PE Samples UPX/"
files = os.listdir(files_path)
file_paths = [files_path + x for x in files]

cmd = "upx -d"

for path in file_paths:
    cmd2 = f'{cmd} "{path}"'
    process = Popen(cmd2, stdout=PIPE, stderr=PIPE, shell=True)
    output, errors = process.communicate()

    # Check for errors in output
    if "error" in output.decode() or errors:
        print(f"Error processing {path}: {errors.decode()}")
    else:
        print(f"Success: {output.decode()}")
```

Kết quả:

```
                       Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    120320 <-     75776   62.98%    win32/pe     EhStorAuthn.exe

Unpacked 1 file.

Success:                      Ultimate Packer for eXecutables
                       Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     81408 <-     76288   93.71%    win32/pe     eventvwr.exe

Unpacked 1 file.

Success:                      Ultimate Packer for eXecutables
                       Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     34304 <-     21504   62.69%    win32/pe     eventcreate.exe

Unpacked 1 file.

Success:                      Ultimate Packer for eXecutables
                       Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020
```

Sau đó tiến hành đóng gói lại, kết quả nhận được như sau:

```
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     37376 ->     21504   57.53%    win64/pe     klist.exe

Packed 1 file.

Success:                      Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    288768 ->    121344   42.02%    win32/pe     imjpuexc.exe

Packed 1 file.

Success:                      Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    123904 ->     49152   39.67%    win32/pe     ieUnatt.exe

Packed 1 file.

Success:                      Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    140288 ->     89088   63.50%    win32/pe     IMESEARCH.EXE
```

Câu 7: Cho biết kết quả đánh giá (Xây dựng bộ phân loại packer)

Đọc tất cả tập tin cần phân tích và gán nhãn cho chúng:

```
[ ]  import os
     from os import listdir

     directories_with_labels = [
         ("Benign PE Samples", 0),
         ("Benign PE Samples UPX", 1),
         ("Benign PE Samples Amber", 2)
     ]
     list_of_samples= []
     labels = []
     for dataset_path, label in directories_with_labels:
       samples = [f for f in listdir(dataset_path)]
       for file in samples:
         file_path = os.path.join(dataset_path, file)
         list_of_samples.append(file_path)
         labels.append(label)
```

Phân ra train test:

```
from sklearn.model_selection import train_test_split

samples_train, samples_test, labels_train, labels_test = train_test_split(
    list_of_samples, labels, test_size=0.3, stratify=labels, random_state=11
)
```

Import thư viện cần thiết để trích xuất N-grams:

```
import collections
from nltk import ngrams
import numpy as np
```

Định nghĩa hàm sử dụng trích xuất N-grams:

```
[ ]  def read_file(file_path):
       """Reads in the  binary sequence of a binary file"""
       with open(file_path, "rb") as binary_file:
         data = binary_file.read()
       return data

     def byte_sequence_to_Ngrams(byte_sequence, N):
       Ngrams = ngrams(byte_sequence, N)
       return list(Ngrams)

     def extract_Ngram_counts(file, N):
       """Takes a binary file and outputs the N-grams counts of its binary sequence"""
       filebyte_sequence = read_file(file)
       file_Ngrams = byte_sequence_to_Ngrams(filebyte_sequence, N)
       return collections.Counter(file_Ngrams)

     def featurize_sample(sample, K1_most_frerquent_Ngrams_list):
       """Takes a sample and produces a feature vector.
           The features are the counts of the K1 N-grams we've selected."""
       K1 = len(K1_most_frequent_Ngrams_list)
       feature_vector = K1 * [0]
       file_Ngrams = extract_Ngram_counts(sample, N)
       for i in range(K1):
         feature_vector[i] = file_Ngrams[K1_most_frequent_Ngrams_list[i]]
       return feature_vector
```

Chọn N-grams mong muốn:

```
N = 2
total_Ngram_count = collections.Counter([])
for file in samples_train:
  total_Ngram_count += extract_Ngram_counts(file, N)
K1 = 100
K1_most_common_Ngrams = total_Ngram_count.most_common(K1)
K1_most_common_Ngrams_list = [x[0] for x in K1_most_common_Ngrams]
```

Thiết lập thuộc tính để huấn luyện:

```
Ngram_features_list_train = []
y_train = []
for i in range(len(samples_train)):
  file = samples_train[i]
  NGram_features = featurize_sample(file, K1_most_common_Ngrams_list)
  Ngram_features_list_train.append(NGram_features)
  y_train.append(labels_train[i])
X_train = Ngram_features_list_train
```

Huấn luyện mô hình random forest trên tập train:

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100)
clf = clf.fit(X_train, y_train)
```

Thiết lập thuộc tính cho tập test:

```
[ ] Ngram_features_list_test = []
    y_test = []
    for i in range(len(samples_test)):
      file = samples_test[i]
      NGram_features = featurize_sample(file, K1_most_common_Ngrams_list)
      Ngram_features_list_test.append(NGram_features)
      y_test.append(labels_test[i])
    X_test = Ngram_features_list_test
```

Sử dụng bộ phân loại được đào tạo để dự đoán trên bộ test và đánh đánh giá hiệu suất bằng confusion matrix:

```
y_pred = clf.predict(X_test)
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

```
array([[ 0, 26,  0],
       [ 0, 60,  0],
       [ 0, 23,  0]])
```

```
[ ] from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, f1_score, precision_score

    print("Confusion matrix:\n %s" % confusion_matrix(y_test, y_pred))

    # Additional metrics (Not required)
    print("Accurcy: %s" %(accuracy_score(y_test, y_pred)))
```

```
Confusion matrix:
 [[ 0 26  0]
 [ 0 60  0]
 [ 0 23  0]]
Accurcy: 0.5504587155963303
```

Câu 8: Cho biết kết quả đánh giá mẫu mới trong việc đánh lừa bộ nhận diện (MalGAN - Tạo phần mềm độc hại)