

# BÁO CÁO THỰC HÀNH

Môn học: Quản trị mạng và hệ thống

**Lab 2: Machine Learning based Malware Detection**

GVHD: Nguyễn Hữu Quyền

## 1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT522.021.ATCL.2- Nhóm 3

STT	Họ và tên	MSSV	Email
1	Nguyễn Ngọc Trà My	21520353	<a href="mailto:21520353@gm.uit.edu.vn">21520353@gm.uit.edu.vn</a>
2	Bùi Hoàng Trúc Anh	21521817	<a href="mailto:21521817@gm.uit.edu.vn">21521817@gm.uit.edu.vn</a>
3	Lê Hoàng Oanh	21521253	<a href="mailto:21521253@gm.uit.edu.vn">21521253@gm.uit.edu.vn</a>
4	Huỳnh Minh Tân Tiến	21521520	21521520@gm.uit.edu.vn

## 2. NỘI DUNG THỰC HIỆN:

STT	Công việc	Kết quả tự đánh giá
1	Yêu cầu 1	100%
2	Yêu cầu 2	100%
3	Yêu cầu 3	100%
4	Yêu cầu 4	100%
5	Yêu cầu 5	100%
6	Yêu cầu 6	100%
7	Yêu cầu 7	100%
8	Yêu cầu 8	100%

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

# BÁO CÁO CHI TIẾT

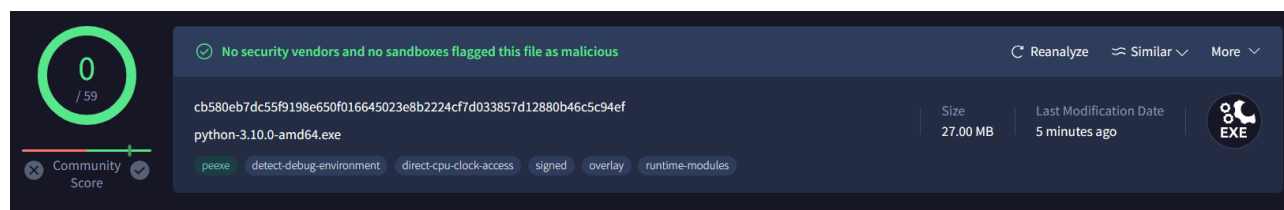
## 1. Sinh viên so sánh kết quả băm với VirusTotal và website Python.

Kết quả thu được:

```
import sys
import hashlib
filename = "/content/python-3.10.0-amd64.exe"
BUF_SIZE = 65536
md5 = hashlib.md5()
sha256 = hashlib.sha256()
with open(filename, "rb") as f:
    while True:
        data = f.read(BUF_SIZE)
        if not data:
            break
        md5.update(data)
        sha256.update(data)
print("MD5: {}".format(md5.hexdigest()))
print("SHA256: {}".format(sha256.hexdigest()))
```

MD5: c3917c08a7fe85db7203da6dcaa99a70  
SHA256: cb580eb7dc55f9198e650f016645023e8b2224cf7d033857d12880b46c5c94ef

Kết quả thu được từ VirusTotal:



Kết quả thu được từ trang chủ của python:

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
<a href="#">Gzipped source tarball</a>	Source release		729e36388ae9a832b01cf9138921b383	23.8 MB	<a href="#">SIG</a>
<a href="#">XZ compressed source tarball</a>	Source release		3e7035d272680f80e3ce4e8eb492d580	17.9 MB	<a href="#">SIG</a>
<a href="#">macOS 64-bit universal2 installer</a>	macOS	for macOS 10.9 and later (updated for macOS 12 Monterey)	8575cc983035ea2f0414e25ce0289ab8	37.9 MB	<a href="#">SIG</a>
<a href="#">Windows installer (64-bit)</a>	Windows	Recommended	c3917c08a7fe85db7203da6dcaa99a70	27.0 MB	<a href="#">SIG</a>
<a href="#">Windows installer (32-bit)</a>	Windows		133aa48145032e341ad2a000cd3bfff50	25.9 MB	<a href="#">SIG</a>
<a href="#">Windows help file</a>	Windows		9d7b80c1c23cfb2cecd63ac4fac9766e	9.1 MB	<a href="#">SIG</a>
<a href="#">Windows embeddable package (64-bit)</a>	Windows		340408540eeff359d5eaf93139ab90fd	8.1 MB	<a href="#">SIG</a>
<a href="#">Windows embeddable package (32-bit)</a>	Windows		dc9d1abc644dd78f5e48edae38c7bc6b	7.2 MB	<a href="#">SIG</a>

Mã hash trùng nhau, file tải về toàn vẹn và khớp data.

## 2. Sinh viên cho biết quả của đoạn code trên

Kết quả trả về bao gồm:

Tên các thư viện



```
D:\Download\Malicious PE Samples 1>scp hostr.exe cuckoo@192.168.30.136:~/Desktop
The authenticity of host '192.168.30.136 (192.168.30.136)' can't be established.
ECDSA key fingerprint is SHA256:F1qy27XymBzdNKRjTSj3YzuLUZH5pBVcWcx16QG3vGs.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Warning: Permanently added '192.168.30.136' (ECDSA) to the list of known hosts.
cuckoo@192.168.30.136's password:
hostr.exe 100%

D:\Download\Malicious PE Samples 1>cd ../Benign PE Samples 1

D:\Download\Benign PE Samples 1>scp hvsirdpclient.exe cuckoo@192.168.30.136:~/Desktop
cuckoo@192.168.30.136's password:
hvsirdpclient.exe 100%
```

Cho cuckoo phân tích file benign là file hvsirdpclient.exe

#### File hvsirdpclient.exe

Summary		<a href="#">Download</a>	<a href="#">Resubmit sample</a>
Size	90.8KB		
Type	data		
MD5	2599040661494a2a9e1ce11200067293		
SHA1	7cc2a762208a274f08326fe437ad75754076168b		
SHA256	8bb2a85b33080e3258f043f62f486cfd6b430fa95ff4169b020c44bd34979fde		
SHA512	<a href="#">Show SHA512</a>		
CRC32	370701BF		
ssdeep	1536:HxTVxpfZzd3ZB3wPhQkGXVJI7tvQm+3eY4I74d6u\Yl1eY8T+TMhJP5ZDwV:5rZZzd3ZhYhQkwVJyvQB/djCLTcJP5i		
Yara	None matched		

#### Information on Execution

Analysis					
Category	Started	Completed	Duration	Routing	Logs
FILE	April 5, 2024, 4:53 a.m.	April 5, 2024, 4:54 a.m.	58 seconds	none	<a href="#">Show Analyzer Log</a> <a href="#">Show Cuckoo Log</a>

#### Network Analysis

[Download pcap](#)

Hosts	2	DNS	4	TCP	0	UDP	26	HTTP	0	ICMP	0	IRC	0	Suricata	Snort
UDP Requests															
192.168.56.101:137		→	192.168.56.255:137												
192.168.56.101:49410		→	224.0.0.252:5355												
192.168.56.101:49690		→	224.0.0.252:5355												
192.168.56.101:52475		→	224.0.0.252:5355												
192.168.56.101:54521		→	224.0.0.252:5355												
192.168.56.101:55982		→	224.0.0.252:5355												
192.168.56.101:56469		→	224.0.0.252:5355												

192.168.56.101:137

→

192.168.56.255:137

16 bytes

32 bytes

48 bytes

64 bytes

plaintexthex

00000000: c11c 2910 0001 0000 0000 0001 2045 5045 ..).....EPE

00000010: 4245 4f45 4943 4e46 4145 4443 4143 4143 BE0E1CFAEDCACAC

00000020: 4143 4143 4143 4143 4143 4143 4100 0020 ACACACACACACA...

00000030: 0001 c00c 0020 0001 0004 93c0 0006 0000 ..Be

00000040: c0a8 3865

192.168.56.101:137

→

192.168.56.255:137

16 bytes

32 bytes

48 bytes

64 bytes

plaintexthex

00000000: c11c 2910 0001 0000 0000 0001 2045 5045 ..).....EPE

00000010: 4245 4f45 4943 4e46 4145 4443 4143 4143 BE0E1CFAEDCACAC

00000020: 4143 4143 4143 4143 4143 4143 4100 0020 ACACACACACACA...

00000030: 0001 c00c 0020 0001 0004 93c0 0006 0000 ..Be

00000040: c0a8 3865

Cho cuckoo phân tích file malicious là file hostr.exe

File hostr.exe

Summary

Download

Resubmit sample

Size	105.0KB
Type	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
MD5	5a559b6d223c79f3736dc52794636cfd
SHA1	5c4676b37fcd49990d21960a2df57af72ceef29a
SHA256	6f201afc797370ac6e33fafec41a794a2eb44c1bfd7d9079e3633ebe7bbb41e1
SHA512	<div>Show SHA512</div>
CRC32	3006EDD4
ssdeep	1536:aDYEasJqkUssX0cfaAJzYU4r/1CbSYlIePDVFkhgIJZH:aasJjUff0derYRH
Yara	None matched

Information on Execution

Analysis

Category	Started	Completed	Duration	Routing	Logs
FILE	April 5, 2024, 4:37 a.m.	April 5, 2024, 4:41 a.m.	205 seconds	none	<div>Show Analyzer Log</div> <div>Show Cuckoo Log</div>

cuckoo

DashboardRecentPendingSearch

Static Analysis

Static AnalysisStringsAntivirusIRMA

PE Compile Time

2023-10-06 07:24:17

PE Imphash

f34d57204577a0bdf0ceec316c175a744

Version Infos

Translation

0x0000 0x04b0

LegalCopyright

Assembly Version

0.0.0.0

InternalName

max.exe

FileVersion

0.0.0.0

ProductVersion

0.0.0.0

FileDescription

OriginalFilename

max.exe

Sections

Name	Virtual Address	Virtual Size	Size of Raw Data	Entropy
.text	0x00002000	0x000035e4	0x00003600	5.30403703612
.pdata	0x00006000	0x00012112	0x00012200	5.35330940155
.rsrc	0x0001a000	0x00004550	0x00004600	7.4789025461
.reloc	0x00020000	0x0000000c	0x00000200	0.0815394123432

Resources

Name	Offset	Size	Language	Sub-language	File type
SIMON	0x0001a0b4	0x00004268	LANG_NEUTRAL	SUBLANG_NEUTRAL	gzip compressed data, max speed, from FAT filesystem (MS-DOS, OS/2, NT)
RT_VERSION	0x0001a31c	0x00000234	LANG_NEUTRAL	SUBLANG_NEUTRAL	data

Imports

Library mscorcore.dll

0x402090 - CorExeMain

#### 4. Tương tự sinh viên hãy làm các câu truy vấn về Python và Powershell Powershell

```

▶ username = "tantien-hmtt"
password = "ghp_uCKp4ZSXC40PrXXu6IyKQPlv5OGdaX4S810H" # .
target_dir = "/content/drive/MyDrive/lab_dataset/lab2/PowerShellSamples"
repositories = github.search_repositories(query="language:powershell")
n = 5
i = 0
count = 0

for repo in repositories:
    reponame = repo.name
    target_dir_of_repo = target_dir + "/" + reponame
    print(reponame)
    try:
        os.mkdir(target_dir_of_repo)
        i += 1
        contents = repo.get_contents("")

        while len(contents) > 1:
            file_content = contents.pop(0)
            if file_content.type == "dir":
                contents.extend(repo.get_contents(file_content.path))
            else:
                st = str(file_content)
                filename = st.split('\"')[1].split('\"')[0]
                extension = filename.split(".")[1]
                if extension == "ps1": # --> py, ps1, js
                    filecontents = repo.get_contents(file_content.path)
                    file_data = base64.b64decode(filecontents.content)
                    filename = filename.split("/")[-1]
                    file_out = open(target_dir_of_repo + "/" + filename, "wb")
                    file_out.write(file_data)

    except:
        pass
    if i == n:
        break

```

Kết quả

```
... core
Scoop
Windows10Debloater
WSL
PowerSploit
winutil
SpotX
blazor
BloodHound
runner-images
nishang
```

### Python


```
username = "tantien-hmtt"
password = "ghp_uCKp4ZSXC40PrXXu6IyKQPlv50GdaX4S810H" # .
target_dir = "/content/drive/MyDrive/lab_dataset/lab2/PythonSamples"
repositories = github.search_repositories(query="language:python")
n = 5
i = 0
count = 0
```

```
for repo in repositories:
    reponame = repo.name
    target_dir_of_repo = target_dir + "/" + reponame
    print(reponame)
    try:
        os.mkdir(target_dir_of_repo)
        i += 1
        contents = repo.get_contents("")

        while len(contents) > 1:
            file_content = contents.pop(0)
            if file_content.type == "dir":
                contents.extend(repo.get_contents(file_content.path))
            else:
                st = str(file_content)
                filename = st.split('')[1].split('')[0]
                extension = filename.split(".")[1]
                if extension == "ps1": # --> py, ps1, js
                    filecontents = repo.get_contents(file_content.path)
                    file_data = base64.b64decode(filecontents.content)
                    filename = filename.split("/")[1]
                    file_out = open(target_dir_of_repo + "/" + filename, "wb")
                    file_out.write(file_data)

    except:
        pass
    if i == n:
        break
```

Kết quả



```
public-apis
awesome-python
Python-100-Days
youtube-dl
HelloGitHub
```

## 5. Sinh viên cho biết quả của đoạn code trên



```
[53] 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
```

```
▶ javascript_path = "/content/drive/MyDrive/lab_dataset/lab2/JavascriptSamples"
  python_path = "/content/drive/MyDrive/lab_dataset/lab2/PythonSamples"
  powershell_path = "/content/drive/MyDrive/lab_dataset/lab2/PowerShellSamples/"
```

```
▶ corpus = []
  labels = []
  file_types_and_labels = [(javascript_path, -1), (python_path, 0), (powershell_path, 1)]
```

```
for files_path, label in file_types_and_labels:
    folder = os.listdir(files_path)
    for subfolder in folder:
        print("==== USING FOLDER: %s =====" % subfolder)
        path = files_path + subfolder
        for file in os.listdir(path):
            print("- Running at file: %s" % (path + '/' + file))
            data = ''

            try:
                with open(f'{path}/{file}', 'r') as myfile:
                    data = myfile.read().replace('\n', '')

            except:
                pass
            data = str(data)
            corpus.append(data)
            labels.append(label)
```

```
▶ X_train, X_test, y_train, y_test = train_test_split(corpus, labels, test_size=0.33, random_state=11)
  text_clf = Pipeline(
      [
          ("vect", HashingVectorizer(input="content", ngram_range=(1,3))),
          ("tfidf", TfidfTransformer(use_idf=True)),
          ("rf", RandomForestClassifier(class_weight="balanced")),
      ]
  )
```

```
[ ] text_clf.fit(x_train, y_train)

Pipeline(steps=[('vect', HashingVectorizer(ngram_range=(1, 3))),
                 ('tfidf', TfidfTransformer()),
                 ('rf', RandomForestClassifier(class_weight='balanced'))])

[ ] y_test_pred = text_clf.predict(X_test)
    print(accuracy_score(y_test, y_test_pred))
    print(confusion_matrix(y_test, y_test_pred))

0.958041958041958
[[ 48   2   2]
 [  0  63   1]
 [  0   7 163]]
```

Kết quả của đoạn code là để phân loại các loại tập tin JavaScript, Python và PowerShell sử dụng mô hình RandomForestClassifier trong scikit-learn

## 6. Sinh viên cho biết quả của đoạn code trên

Đầu tiên ta kiểm tra hash và độ giống nhau của 4 string sử dụng một thuật toán băm ssdeep.

```
ubuntu@ubuntu-virtual-machine:~$ python3 check.py
3:f4oo8MRwRJFGW1gC6uWvPMFSL+JuBF8BSnJi:f4kPvtHM0byFtQ
3:f4oo8MRwRJFGW1gC6uWvPMFSL+JuBF8BS+EFECJi:f4kPvtHM0byFIsJQ
3:f4oo8MRwRJFGW1gC6uWvPMFSL+JuBF8BS6:f4kPvtHM0byF0
3:60QKZ+4CDTfDaRFKYLVL:ywKDC2mVL
100
36
34
0
```

Ở 4 dòng đầu, ta thấy 4 file cho ra mã hash hoàn toàn khác nhau vì nội dung khác biệt nhau.

Ở 4 dòng sau, kết quả so sánh hash1-hash1 là giống nhau hoàn toàn, hash1-hash2 và hash1-hash3 là gần giống và kết quả của hash1-hash4 là hoàn toàn khác.

Tiếp theo tạo một bản sao từ tập tin python-3.10.0-amd64.exe thành python 3.10.0-amd64-fake.exe bằng cách thêm vài null bytes bằng lệnh truncate

Dùng hexdump để xem sự khác nhau giữa hai tập tin trước và sau

```

ubuntu@ubuntu-virtual-machine:~/Downloads$ hexdump -C python-3.10.0-amd64.exe | tail -5
01b010e0 10 9c 34 66 02 d3 51 8c b1 64 19 f3 55 12 0e 74 |..4f..Q..d..U..t|
01b010f0 38 71 4c 2e 1c db 44 d4 f3 81 31 a5 9c 2e c6 06 |8qL...D...1....|
01b01100 4f 33 c6 8a 9a 5e 16 52 8c 4b 55 10 2b cd 45 61 |03...^..R..KU..+..Ea|
01b01110 a5 00 00 00 00 00 00 00 |.....|
01b01118
ubuntu@ubuntu-virtual-machine:~/Downloads$ hexdump -C python-3.10.0-amd64-fake.exe | tail -5
01b010e0 10 9c 34 66 02 d3 51 8c b1 64 19 f3 55 12 0e 74 |..4f..Q..d..U..t|
01b010f0 38 71 4c 2e 1c db 44 d4 f3 81 31 a5 9c 2e c6 06 |8qL...D...1....|
01b01100 4f 33 c6 8a 9a 5e 16 52 8c 4b 55 10 2b cd 45 61 |03...^..R..KU..+..Ea|
01b01110 a5 00 00 00 00 00 00 00 |.....|
01b01119

```

Chạy code check độ giống ta có kết quả được xem là giống nhau 100%

```

ubuntu@ubuntu-virtual-machine:~/Downloads$ python3 newCheck.py
100

```

## 7. Sinh viên cho biết quả của đoạn code trên

Kết quả khi lấy N-grams với từng phương pháp

Frequency:

```

X_top_K2_freq = X[:, :K2]
X_top_K2_freq
array([[ 10935, 4673, 7, 610, 26, 0, 13,
        [ 15237, 2604, 866, 1848, 1332, 1343, 630,
        [ 4963, 282, 88, 129, 222, 573, 120,
        [ 36882, 1921, 6191, 6527, 2784, 0, 1435,
        [ 140825, 11831, 7, 1350, 25, 2, 3,
        [ 527, 670, 8,
        [ 18655, 1518, 3, 1176, 173, 1, 1,
        [ 265, 1621, 5,
        [ 17320, 1851, 34, 770, 43, 1, 3,
        [ 295, 402, 21,
        [ 7159, 2219, 3, 1136, 29, 0, 0,
        [ 350, 949, 5,
        [ 71210, 5507, 2, 259, 10, 1, 1,
        [ 210, 225, 2,
        [ 15222, 699, 0, 186, 9, 1, 0,
        [ 202, 191, 2,
        [ 19249, 788, 2849, 2681, 2903, 1, 1366,
        [ 319, 548, 531,
        [ 10478, 469, 732, 338, 479, 0, 303,
        [ 131, 485, 377,
        [ 7719, 274, 628, 384, 376, 0, 185,

```

Mutual information:

```

mi_selector = SelectKBest(mutual_info_classif, k=K2)
X_top_K2_mi = mi_selector.fit_transform(X, y)
X_top_K2_mi
array([[ 10, 2, 2, 3, 12, 35, 2, 14, 0, 0],
        [ 0, 1, 8, 13, 21, 13, 10, 8, 140, 0],
        [ 0, 0, 0, 2, 1, 0, 0, 1, 1, 0],
        [ 5, 13, 36, 17, 144, 32, 2, 48, 92, 0],
        [ 162, 8, 3, 5, 4, 23, 15, 51, 2, 736],
        [ 0, 3, 0, 11, 2, 28, 2, 7, 1, 0],
        [ 38, 2, 2, 4, 2, 13, 2, 17, 1, 0],
        [ 1, 5, 0, 17, 5, 17, 2, 11, 0, 0],
        [ 81, 4, 1, 0, 3, 7, 8, 1, 368],
        [ 7, 0, 2, 2, 1, 2, 5, 4, 1, 8],
        [ 2, 7, 26, 11, 29, 17, 8, 21, 49, 0],
        [ 5, 4, 2, 1, 8, 6, 3, 2, 3, 1],
        [ 0, 3, 2, 0, 10, 5, 2, 1, 7, 0],
        [ 3, 24, 45, 35, 65, 43, 6, 37, 69, 2],
        [ 0, 0, 4, 2, 14, 1, 19, 2, 3, 0],
        [ 6, 113, 210, 129, 218, 179, 3, 93, 227, 5],
        [ 16, 76, 239, 122, 228, 225, 24, 53, 933, 52],
        [ 1, 4, 8, 2, 2, 2, 11, 3, 10, 1],
        [ 0, 4, 16, 0, 8, 16, 0, 4, 6, 0],
        [ 0, 5, 3, 2, 10, 6, 6, 1, 18, 0],
        [ 0, 9, 23, 8, 49, 21, 15, 9, 18, 0],
        [ 18, 552, 515, 217, 396, 488, 46, 319, 157, 55],
        [ 12, 8, 16, 20, 14, 40, 11, 7, 15, 0],
        [ 1, 1, 1, 0, 8, 12, 11, 2, 1, 0],
        [ 1, 2, 6, 1, 9, 2, 8, 4, 7, 0],

```

Chi-squared:

```

chi2_selector = SelectKBest(chi2, k=k2)
X_top_k2_ch2 = chi2_selector.fit_transform(X, y)
X_top_k2_ch2

array([[ 615, 433, 574, 335, 46, 552, 84, 4, 7,
        [ 171, 32, 125, 6, 11, 46, 1, 4, 0,
        [ 19, 0, 10, 1, 3, 6, 0, 0, 0,
        [ 436, 447, 80, 8, 623, 12, 1, 6, 0,
        [ 75, 192, 165, 72, 57, 54, 19, 3, 27,
        [ 149, 222, 147, 90, 54, 121, 40, 3, 24,
        [ 72, 127, 126, 48, 30, 81, 27, 5, 14,
        [ 62, 191, 218, 104, 57, 61, 11, 4, 11,
        [ 16, 22, 42, 9, 5, 13, 5, 1, 7,
        [ 9, 26, 26, 7, 3, 14, 1, 0, 1,
        [ 130, 115, 78, 6, 71, 71, 1, 0, 1,
        [ 131, 10, 19, 1, 8, 2, 2, 3, 0,
        [ 90, 6, 12, 1, 7, 2, 1, 1, 0,
  
```

## 8. Sinh viên hoàn thành các bước trên

### B1. Tạo list các mẫu và gán nhãn cho chúng.

```

[74] import os
      from os import listdir

[75] directories_and_labels = [("/content/drive/MyDrive/ML/(Lab2)Dataset/Benign PE Samples", 0), ("/content/drive/MyDrive/ML/(Lab2)Dataset/Malicious PE Samples", 1)]
      list_of_samples = []
      labels = []
      N_spec = 2 # For N-grams

for dataset_path, label in directories_and_labels:
    samples = [f for f in listdir(dataset_path)]
    for sample in samples:
        file_path = os.path.join(dataset_path, sample)
        list_of_samples.append(file_path)
        labels.append(label)

[77] print(list_of_samples[:5]) # Get the first 5 samples
      print(list_of_samples[len(list_of_samples)-5: len(list_of_samples)]) # Get the last 5 samples

['/content/drive/MyDrive/ML/(Lab2)Dataset/Benign PE Samples/junction.exe', '/content/drive/MyDrive/ML/(Lab2)Dataset/Benign PE Samples/ls.exe', '/content/drive/MyDrive/ML/(Lab2)Dataset/Benign PE Samples/HxCalendarAppImm.exe', '/content/drive/MyDrive/ML/(Lab2)Dataset/Benign PE Samples/IEChooser.exe', '/content/drive/MyDrive/ML/(Lab2)Dataset/Benign PE Samples/IEChooser.exe']

print(labels[:5]) # Get the first 5 samples
print(labels[len(list_of_samples)-5: len(list_of_samples)]) # Get the last 5 samples

[0, 0, 0, 0, 0]
[0, 0, 0, 0, 1]
  
```

### B2. Chia dữ liệu train-test

```

# Split the dataset into training and testing sets
samples_train, samples_test, target_train, target_test = train_test_split(
    list_of_samples, labels, test_size=0.2, random_state=42)

# Ensure at least one malicious file is in the testing set
# Check if there's at least one malicious file in the testing set
malicious_in_testing = False
for sample, label in zip(samples_test, target_test):
    if label == 1: # Malicious
        malicious_in_testing = True
        break

# If no malicious file is in the testing set, move one from training to testing
if not malicious_in_testing:
    for i, label in enumerate(target_train):
        if label == 1: # Malicious
            samples_test.append(samples_train.pop(i))
            target_test.append(target_train.pop(i))
            break

print("Training samples:", len(samples_train))
print("Testing samples:", len(samples_test))
print("Number of malicious samples in testing:", sum(target_test))

Training samples: 68
Testing samples: 19
Number of malicious samples in testing: 1
  
```

## B3. Các hàm lấy thuộc tính

```

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

def read_file(file_path):
    with open(file_path, "rb") as bin_file:
        data = bin_file.read()
    return data

def byte_seq_to_Ngrams(byte_seq, N_par):
    Ngrams_par = ngrams(byte_seq, N_par)
    return list(Ngrams_par)

def bin_file_to_Ngrams_count(file_path, N_par):
    file_seq = read_file(file_path)
    file_Ngrams = byte_seq_to_Ngrams(file_seq, N_par)
    return collections.Counter(file_Ngrams)

def get_Ngrams_features_from_samples(sample, K1_most_freq_Ngrams_list):
    K1 = len(K1_most_freq_Ngrams_list)
    feature_vector = K1 * [0]
    file_Ngrams = bin_file_to_Ngrams_count(sample, N_spec)
    for i in range(K1):
        feature_vector[i] = file_Ngrams[K1_most_freq_Ngrams_list[i]]
    return feature_vector

def preprocess_imports(list_of_DLLs):
    """ Normalize the name of the imports of a PE file. """
    temp = [x.decode().split(".")[0].lower() for x in list_of_DLLs] # View the transforming of below example
    return " ".join(temp)

def get_imports(pe):
    """ Get a list of the imports of a PE file """
    list_of_imports = []
    for entry in pe.DIRECTORY_ENTRY_IMPORT:
        list_of_imports.append(entry.dll)
    return preprocess_imports(list_of_imports)

def get_section_names(pe):
    """ Get a list of the section names of a PE file """
    list_of_sections = []
    for sect in pe.sections:
        normalized_name = sect.Name.decode().replace("\x00", "").lower()
        list_of_sections.append(normalized_name)
    return " ".join(list_of_sections)

```

## B4. Chọn 100 thuộc tính phổ biến với 2-grams

```

Ngrams_count_all = collections.Counter([])
for sample in samples_train:
    Ngrams_count_all += bin_file_to_Ngrams_count(sample, N_spec)
K1 = 100
K1_most_common_Ngrams = Ngrams_count_all.most_common(K1)
K1_most_common_Ngrams_list = [x[0] for x in K1_most_common_Ngrams]

```

## B5. Trích xuất số lượng N-grams count, section names, imports và số lượng sections của mỗi mẫu trong train-test.

```

imports_corpus_train = []
num_sect_train = []
sect_name_train = []
Ngram_feat_list_train = []

y_train = []
for i in range(len(samples_train)):
    sample = samples_train[i]
    try:
        # Get all required parameters with predefined functions
        Ngram_features = get_Ngrams_features_from_samples(sample, K1_most_common_Ngrams_list)
        pe = pefile.PE(sample)
        imports = get_imports(pe)
        n_sections = len(pe.sections)
        sec_names = get_section_names(pe)

        # Put above value into lists
        imports_corpus_train.append(imports)
        num_sect_train.append(n_sections)
        sect_name_train.append(sec_names)
        Ngram_feat_list_train.append(Ngram_features)

        # Target train
        y_train.append(target_train[i])
    except Exception as e:
        print(sample + ":")
        print(e)

```

B6. Sử dụng hàm băm tfidf để chuyển imports, section names từ văn bản thành dạng số

```

from sklearn.feature_extraction.text import HashingVectorizer, TfidfTransformer
from sklearn.pipeline import Pipeline
imports_featurizer = Pipeline(
    [
        ("vect", HashingVectorizer(input = "content", ngram_range=(1,2))),
        ("tfidf", TfidfTransformer(use_idf = True)),
    ]
)
sect_name_featurizer = Pipeline(
    [
        ("vect", HashingVectorizer(input = "content", ngram_range= (1,2))),
        ("tfidf", TfidfTransformer(use_idf = True))
    ]
)
imports_corpus_train_transformed = imports_featurizer.fit_transform(imports_corpus_train)
sect_name_train_transformed = sect_name_featurizer.fit_transform(sect_name_train)

```

B7. Kết hợp các vector thuộc tính thành 1 mảng.

```

from scipy.sparse import hstack, csr_matrix
X_train = hstack(
    [
        Ngram_feat_list_train,
        imports_corpus_train_transformed,
        sect_name_train_transformed,
        csr_matrix(num_sect_train).transpose(),
    ]
)

```

B8. Ta huấn luyện bằng phân loại Random Forest cho tập train

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

B9. Thu thập các thuộc tính của tập test, giống như tập huấn luyện

```
import_corpus_test = []  
num_sect_test = []  
sect_names_test = []  
Ngram_feat_list_test = []  
  
y_test = []  
for i in range(len(samples_test)):  
    test = samples_test[i]  
    try:  
        # Get all required parameters with predefined functions  
        # The input when getting N-grams features is still "sample"  
        Ngram_features = get_Ngrams_features_from_samples(sample, K1_most_common_Ngrams_list)  
        pe = pefile.PE(test) # Get test PE file  
        imports = get_imports(pe)  
        n_sections = len(pe.sections)  
        sec_names = get_section_names(pe)  
  
        # Put above value into lists  
        import_corpus_test.append(imports)  
        num_sect_test.append(n_sections)  
        sect_names_test.append(sec_names)  
        Ngram_feat_list_test.append(Ngram_features)  
  
        # Target train  
        y_test.append(target_test[i])  
    except Exception as e:  
        print(sample + ":", "  
        print(e)  
y_test
```

```
/content/drive/MyDrive/(Lab2)Dataset/Benign PE Samples/icsunattend.exe:  
'DOS Header magic not found.'  
/content/drive/MyDrive/(Lab2)Dataset/Benign PE Samples/icsunattend.exe:  
'DOS Header magic not found.'  
/content/drive/MyDrive/(Lab2)Dataset/Benign PE Samples/icsunattend.exe:  
'DOS Header magic not found.'  
/content/drive/MyDrive/(Lab2)Dataset/Benign PE Samples/icsunattend.exe:  
'DOS Header magic not found.'  
/content/drive/MyDrive/(Lab2)Dataset/Benign PE Samples/icsunattend.exe:  
'DOS Header magic not found.'  
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

B10. Ta chuyển đổi vector từ thuộc tính test, và kiểm tra kết quả của trình phân loại.

```
import_corpus_test_transformed = imports_featurizer.transform(import_corpus_test)
sect_names_test_transformed = imports_featurizer.transform(sect_names_test)
X_test = hstack(
    [
        Ngram_feat_list_test,
        import_corpus_test_transformed,
        sect_names_test_transformed,
        csr_matrix(num_sect_test).transpose()
    ]
)
print("The score of our classifier is as follow: ")
print(clf.score(X_test, y_test))

The score of our classifier is as follow:
0.9285714285714286
```