

✓ Lab 2 - Machine Learning based Malware Detection

21520155 - Nguyễn Triệu Thiên Bảo

21521195 - Trần Lê Minh Ngọc

Download các thư viện và công cụ cần thiết

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
!sudo apt-get install automake libtool make gcc pkg-config
```

```

[+] Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
automake is already the newest version (1:1.16.5-1.3).
automake set to manually installed.
gcc is already the newest version (4:11.2.0-1ubuntu1).
gcc set to manually installed.
make is already the newest version (4.3-4.1build1).
make set to manually installed.
pkg-config is already the newest version (0.29.2-1ubuntu3).
Suggested packages:
  libtool-doc gcj-jdk
The following NEW packages will be installed:
  libtool
0 upgraded, 1 newly installed, 0 to remove and 45 not upgraded.
Need to get 164 kB of archives.
After this operation, 1,227 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 libtool all 2.4.6-15build2 [164 kB]
Fetched 164 kB in 0s (1,857 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package libtool.
(Reading database ... 121753 files and directories currently installed.)
Preparing to unpack .../libtool_2.4.6-15build2_all.deb ...
Unpacking libtool (2.4.6-15build2) ...
Setting up libtool (2.4.6-15build2) ...
Processing triggers for man-db (2.10.2-1) ...

```

```
!sudo apt-get install flex bison
```

```

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libfl-dev libfl2
Suggested packages:
  bison-doc flex-doc
The following NEW packages will be installed:
  bison flex libfl-dev libfl2
0 upgraded, 4 newly installed, 0 to remove and 45 not upgraded.
Need to get 1,072 kB of archives.
After this operation, 3,667 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 flex amd64 2.6.4-8build2 [307 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 bison amd64 2:3.8.2+dfsg-1build1 [748 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfl2 amd64 2.6.4-8build2 [10.7 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfl-dev amd64 2.6.4-8build2 [6,236 B]
Fetched 1,072 kB in 0s (7,970 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontE
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:

```

```

Selecting previously unselected package flex.
(Reading database ... 121773 files and directories currently installed.)
Preparing to unpack .../flex_2.6.4-8build2_amd64.deb ...
Unpacking flex (2.6.4-8build2) ...
Selecting previously unselected package bison.
Preparing to unpack .../bison_2%3a3.8.2+dfsg-1build1_amd64.deb ...
Unpacking bison (2:3.8.2+dfsg-1build1) ...
Selecting previously unselected package libfl2:amd64.
Preparing to unpack .../libfl2_2.6.4-8build2_amd64.deb ...
Unpacking libfl2:amd64 (2.6.4-8build2) ...
Selecting previously unselected package libfl-dev:amd64.
Preparing to unpack .../libfl-dev_2.6.4-8build2_amd64.deb ...
Unpacking libfl-dev:amd64 (2.6.4-8build2) ...
Setting up flex (2.6.4-8build2) ...
Setting up libfl2:amd64 (2.6.4-8build2) ...
Setting up bison (2:3.8.2+dfsg-1build1) ...
update-alternatives: using /usr/bin/bison.yacc to provide /usr/bin/yacc (yacc) in auto mode
Setting up libfl-dev:amd64 (2.6.4-8build2) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 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/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

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

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

```

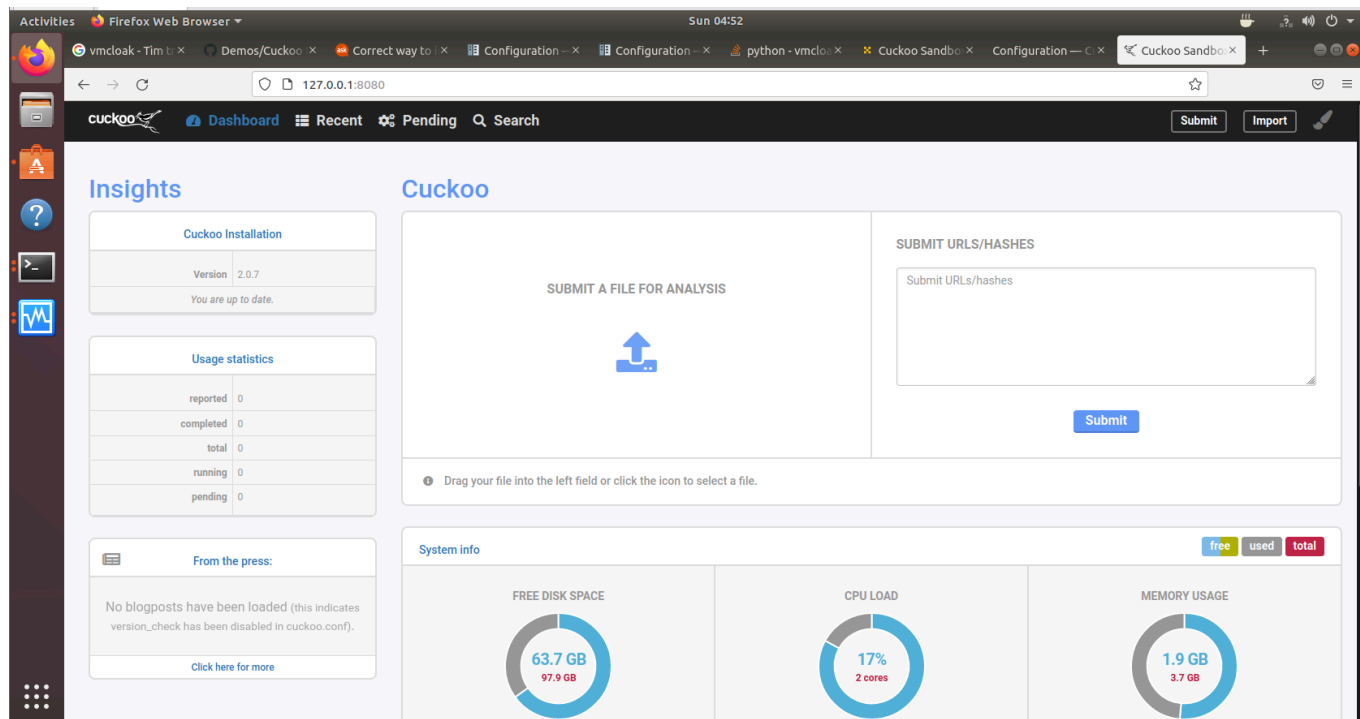
!pip install pefile

```

Collecting pefile
  Downloading pefile-2023.2.7-py3-none-any.whl (71 kB)
    71.8/71.8 kB 1.1 MB/s eta 0:00:00
Installing collected packages: pefile
Successfully installed pefile-2023.2.7

```

1. Phân tích tĩnh mã độc



✓ a) Tính toán hàm băm của một mẫu

```
import sys
import hashlib

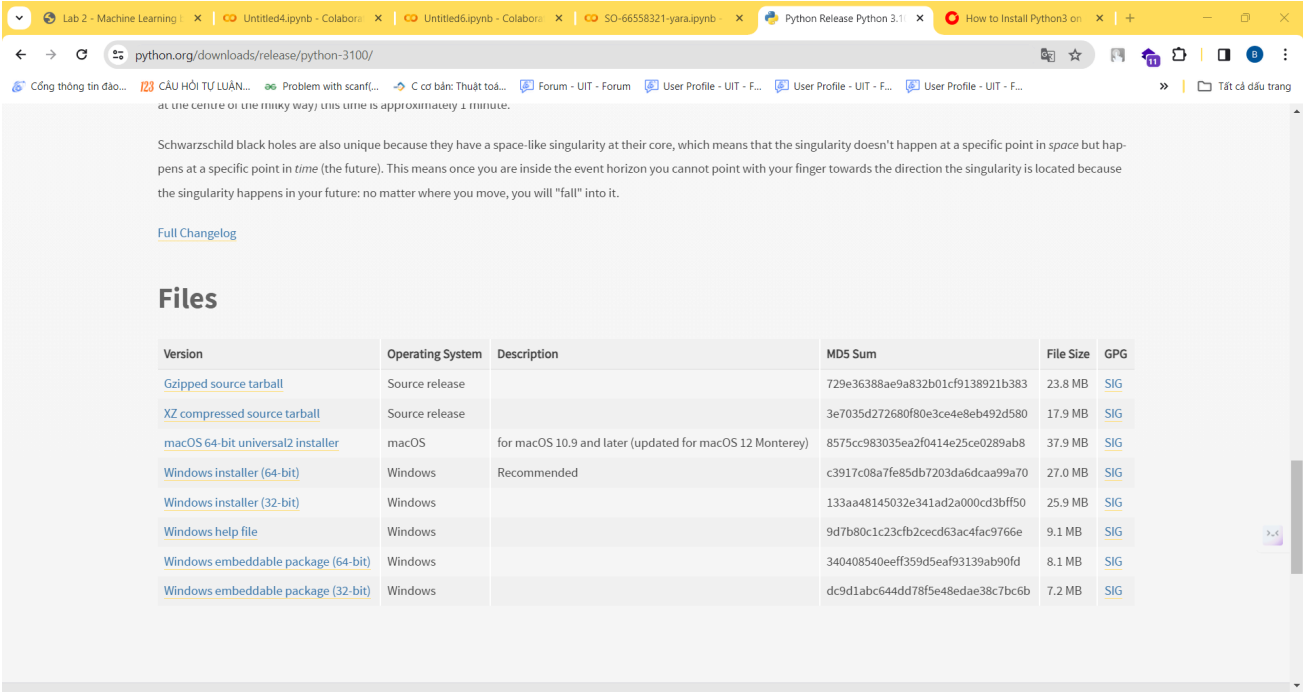
data_path = "/content/drive/MyDrive/NT522.021.ANTT_Lab2/python-3.10.0-amd64.exe"

BUF_SIZE = 65536
md5 = hashlib.md5()
sha256 = hashlib.sha256()
with open(data_path, "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 MD5 trên Website chính thức của Python



Kết quả SHA256 trên trang VirusTotal

cb580eb7dc55f9198e650f016645023e8b2224cf7d033857d12880b46c5c94ef

python-3.10.0-amd64.exe

Size: 27.00 MB | Last Modification Date: 1 hour ago

peexe runtime-modules overlay direct-cpu-clock-access signed detect-debug-environment

0 / 71 Community Score

No security vendors and no sandboxes flagged this file as malicious

Reanalyze Similar More

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Security vendors' analysis

Acronis (Static ML)	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	AliCloud	Undetected
ALYac	Undetected	Antiy-AVL	Undetected
Arcabit	Undetected	Avast	Undetected
AVG	Undetected	Avira (no cloud)	Undetected

=> 2 kết quả bầm đều khớp

✓ b) YARA

B1. Tạo tập tin rules.yara và thêm rule sau để kiểm tra file có phải là pdf hay không

Nếu đúng thì trả về "dummy_rule2" và nếu sai thì trả về "dummy_rule1"

```
GNU nano 7.2 yara.rules
rule is_a_pdf
{
  meta:
    author = "ngoc"
  strings:
    $pdf_magic = {25 50 44 46}
  condition:
    $pdf_magic at 0
}
rule dummy_rule1
{
  meta:
    author = "ngoc"
  condition:
    false
}
rule dummy_rule2
{
  meta:
    author = "ngoc"
  condition:
    true
}
```

B2. Chọn tập tin PDF kiểm tra bằng lệnh yara

Kết quả trả về đúng là file PDF

```
(ngoc@ngoc) - [~/Documents]
$ yara yara.rules Lab2.pdf
is_a_pdf Lab2.pdf
dummy_rule2 Lab2.pdf
```

✓ c) Kiểm tra PE header

```

# B1. Import thư viện pefile và thêm tập tin PE muốn parse
import pefile
desired_file = "/content/drive/MyDrive/NT522.021.ANTT_Lab2/python-3.10.0-amd64.exe"
pe = pefile.PE(desired_file)

# B2. Liệt kê các import của tập tin PE
for entry in pe.DIRECTORY_ENTRY_IMPORT:
    print(entry.dll)
    for imp in entry.imports:
        print("\t", hex(imp.address), imp.name)

    b'ADVAPI32.dll'
        0x44b000 b'RegCloseKey'
        0x44b004 b'RegOpenKeyExW'
        0x44b008 b'OpenProcessToken'
        0x44b00c b'AdjustTokenPrivileges'
        0x44b010 b'LookupPrivilegeValueW'
        0x44b014 b'InitiateSystemShutdownExW'
        0x44b018 b'GetUserNameW'
        0x44b01c b'RegQueryValueExW'
        0x44b020 b'RegDeleteValueW'
        0x44b024 b'CloseEventLog'
        0x44b028 b'OpenEventLogW'
        0x44b02c b'ReportEventW'
        0x44b030 b'ConvertStringSecurityDescriptorToSecurityDescriptorW'
        0x44b034 b'DecryptFileW'
        0x44b038 b'CreateWellKnownSid'
        0x44b03c b'InitializeAcl'
        0x44b040 b'SetEntriesInAclW'
        0x44b044 b'ChangeServiceConfigW'
        0x44b048 b'CloseServiceHandle'
        0x44b04c b'ControlService'
        0x44b050 b'OpenSCManagerW'
        0x44b054 b'OpenServiceW'
        0x44b058 b'QueryServiceStatus'
        0x44b05c b'SetNamedSecurityInfoW'
        0x44b060 b'CheckTokenMembership'
        0x44b064 b'AllocateAndInitializeSid'
        0x44b068 b'SetEntriesInAclA'
        0x44b06c b'SetSecurityDescriptorGroup'
        0x44b070 b'SetSecurityDescriptorOwner'
        0x44b074 b'SetSecurityDescriptorDacl'
        0x44b078 b'InitializeSecurityDescriptor'
        0x44b07c b'RegSetValueExW'
        0x44b080 b'RegQueryInfoKeyW'
        0x44b084 b'RegEnumValueW'
        0x44b088 b'RegEnumKeyExW'
        0x44b08c b'RegDeleteKeyW'
        0x44b090 b'RegCreateKeyExW'
        0x44b094 b'GetTokenInformation'
        0x44b098 b'CryptDestroyHash'
        0x44b09c b'CryptHashData'
        0x44b0a0 b'CryptCreateHash'
        0x44b0a4 b'CryptGetHashParam'
        0x44b0a8 b'CryptReleaseContext'
        0x44b0ac b'CryptAcquireContextW'
        0x44b0b0 b'QueryServiceConfigW'

    b'USER32.dll'
        0x44b35c b'PeekMessageW'
        0x44b360 b'PostMessageW'
        0x44b364 b'IsWindow'
        0x44b368 b'WaitForInputIdle'
        0x44b36c b'PostQuitMessage'
        0x44b370 b'GetMessageW'
        0x44b374 b'TranslateMessage'
        0x44b378 b'MsgWaitForMultipleObjects'
        0x44b37c b'PostThreadMessageW'
        0x44b380 b'GetMonitorInfoW'
        0x44b384 b'MonitorFromPoint'

# B3. Liệt kê các section của tập tin PE
for section in pe.sections:
    print(
        section.Name,
        hex(section.VirtualAddress),
        hex(section.Misc_VirtualSize),
        section.SizeOfRawData,
    )

```

```
# B4. In tất cả thông tin dump từ PE
print(pe.dump_info())
```

d) Featurizing the PE header

✓ 2. Sinh viên cho biết quả của đoạn code sau

```
# B1. Import pefile và hai thư viện
import pefile
from os import listdir
from os.path import isfile, join
directories = ["/content/drive/MyDrive/Benign_PE_Samples", "/content/drive/MyDrive/Malicious_PE_Samples"]

# B2. Định nghĩa hai phương thức thu thập tên của sections và chuẩn hoá chúng.
def get_section_names(pe):
    """Gets a list of section names from a PE file."""
    list_of_section_names = []
    for sec in pe.sections:
        normalized_name = sec.Name.decode().replace("\x00", "").lower()
        list_of_section_names.append(normalized_name)
    return list_of_section_names

# B3. Ta định nghĩa một phương thuận tiện trong tiền xử lý import
def preprocess_imports(list_of_DLLs):
    """Normalize the naming of the imports of a PE file."""
    return [x.decode().split(".")[0].lower() for x in list_of_DLLs]

# B4. Chúng ta định nghĩa hàm thu thập import từ tập tin
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)

# B5. Cuối cùng, duyệt quá tất cả tập tin và tạo danh sách thuộc tính
imports_corpus = []
num_sections = []
section_names = []
for dataset_path in directories:
    samples = [f for f in listdir(dataset_path) if
isfile(join(dataset_path, f))]
    for file in samples:
        file_path = dataset_path + "/" + file
# B6. Ngoài việc thu thập thuộc tính, ta còn thu thập số lượng section của tập tin
try:
    pe = pefile.PE(file_path)
    imports = get_imports(pe)
    n_sections = len(pe.sections)
    sec_names = get_section_names(pe)
    imports_corpus.append(imports)
    num_sections.append(n_sections)
    section_names.append(sec_names)
# B7. Trong trường hợp không parse được tập tin PE, thêm try-catch
except Exception as e:
    print(e)
    print("Unable to obtain imports from " + file_path)

'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/ldp.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/hvsirdpclient.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/hvsirpcd.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/inetinfo.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/InetMgr.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/LxRun.exe
```

```
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/iisrstas.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/lpq.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/lpr.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/LogCollector.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/InspectVhdDialog.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/InspectVhdDialog6.2.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/InspectVhdDialog6.3.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/iissetup.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/InetMgr6.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/iisreset.exe
'DOS Header magic not found.'
Unable to obtain imports from /content/drive/MyDrive/Benign_PE_Samples/ldifde.exe
```

```
print("Imports: %s" % imports_corpus)
print("The number of sections: %s" % num_sections)
print("Sections: %s" % section_names)
```

```
Imports: [['kernel32', 'user32', 'gdi32', 'comdlg32', 'advapi32'], ['cygwin1', 'cygintl-8', 'kernel32'], ['msvc
The number of sections: [4, 10, 8, 7, 5, 5, 5, 5, 5, 7, 7, 6, 6, 7, 7, 6, 6, 6, 7, 7, 6, 6, 6, 5, 5, 5, 5, 5,
Sections: ['.text', '.rdata', '.data', '.rsrc'], ['.text', '.data', '.rdata', '/4', '.pdata', '.xdata', '.bss', '.idata', '.rsrc', '/14
```

Kết quả trả về tên các thư viện imports, số lượng section cũng như tên các section đó.

✓ e) Scraping GitHub cho các loại tập tin đặc biệt

```
!pip install PyGitHub
```

```
Collecting PyGitHub
  Downloading PyGithub-2.3.0-py3-none-any.whl (354 kB)
    354.4/354.4 kB 7.0 MB/s eta 0:00:00
Collecting pynacl>=1.4.0 (from PyGitHub)
  Downloading PyNaCl-1.5.0-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_24_x86_64.whl (856 kB)
    856.7/856.7 kB 33.1 MB/s eta 0:00:00
Requirement already satisfied: requests>=2.14.0 in /usr/local/lib/python3.10/dist-packages (from PyGitHub) (2.31.0)
Collecting pyjwt[crypto]>=2.4.0 (from PyGitHub)
  Downloading PyJWT-2.8.0-py3-none-any.whl (22 kB)
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from PyGitHub) (4.10.0)
Requirement already satisfied: urllib3>=1.26.0 in /usr/local/lib/python3.10/dist-packages (from PyGitHub) (2.0.7)
Collecting Deprecated (from PyGitHub)
  Downloading Deprecated-1.2.14-py2.py3-none-any.whl (9.6 kB)
Requirement already satisfied: cryptography>=3.4.0 in /usr/local/lib/python3.10/dist-packages (from pyjwt[crypto]>=2.4.0->PyGitHub) (42.0.7)
Requirement already satisfied: cffi>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from pynacl>=1.4.0->PyGitHub) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.14.0->PyGitHub) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.14.0->PyGitHub) (3.6)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.14.0->PyGitHub) (2024.2.2)
Requirement already satisfied: wrapt<2,>=1.10 in /usr/local/lib/python3.10/dist-packages (from Deprecated->PyGitHub) (1.14.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.4.1->pynacl>=1.4.0->PyGitHub) (2.22)
Installing collected packages: pyjwt, Deprecated, pynacl, PyGitHub
  Attempting uninstall: pyjwt
    Found existing installation: PyJWT 2.3.0
    Uninstalling PyJWT-2.3.0:
      Successfully uninstalled PyJWT-2.3.0
Successfully installed Deprecated-1.2.14 PyGitHub-2.3.0 pyjwt-2.8.0 pynacl-1.5.0
```

Javascript

```

# B1. Import thư viện PyGithub để gọi API của Github và sử dụng module base64 để mã hoá và giải mã tập tin.
import os
from github import Github
import base64

# B2. Ta phải cung cấp thông tin chứng thực và đưa ra một truy vấn JavaScript trong repository
username = "MN911718"
password = "ghp_li0deOKT6bWco1JASAKoYufq0ILhDR07wRMa"
target_dir = "/content/drive/MyDrive/NT522.O21.ANTT_Lab2/JavaScripts/"
g = Github(username, password)
repositories = g.search_repositories(query="language:javascript")
n = 5
i = 0
# B3. Duyệt danh sách repository trả về
for repo in repositories:
    reponame = repo.name
    target_dir_of_repo = target_dir + "\\" + reponame
    print(reponame)
    try:
# B4. Tạo thư mục lưu trữ
        os.mkdir(target_dir_of_repo)
        i += 1
        contents = repo.get_contents("")
# B5. Ta thêm tất cả các thư mục của repository vào hàng đợi để liệt kê tất cả các tập tin trong thư mục
        while len(contents) > 1:
            file_content = contents.pop(0)
            if file_content.type == "dir":
                contents.extend(repo.get_contents(file_content.path))
            else:
# B6. Nếu kiểm tra một tập tin không phải là thư mục thì kiểm tra phần mở rộng có phải là .js
                st = str(file_content)
                filename = st.split('')[1].split('')[0]
                extension = filename.split(".")[1]
                if extension == "js":
# B7. Nếu là .js thì sẽ ghi ra tập tin
                    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

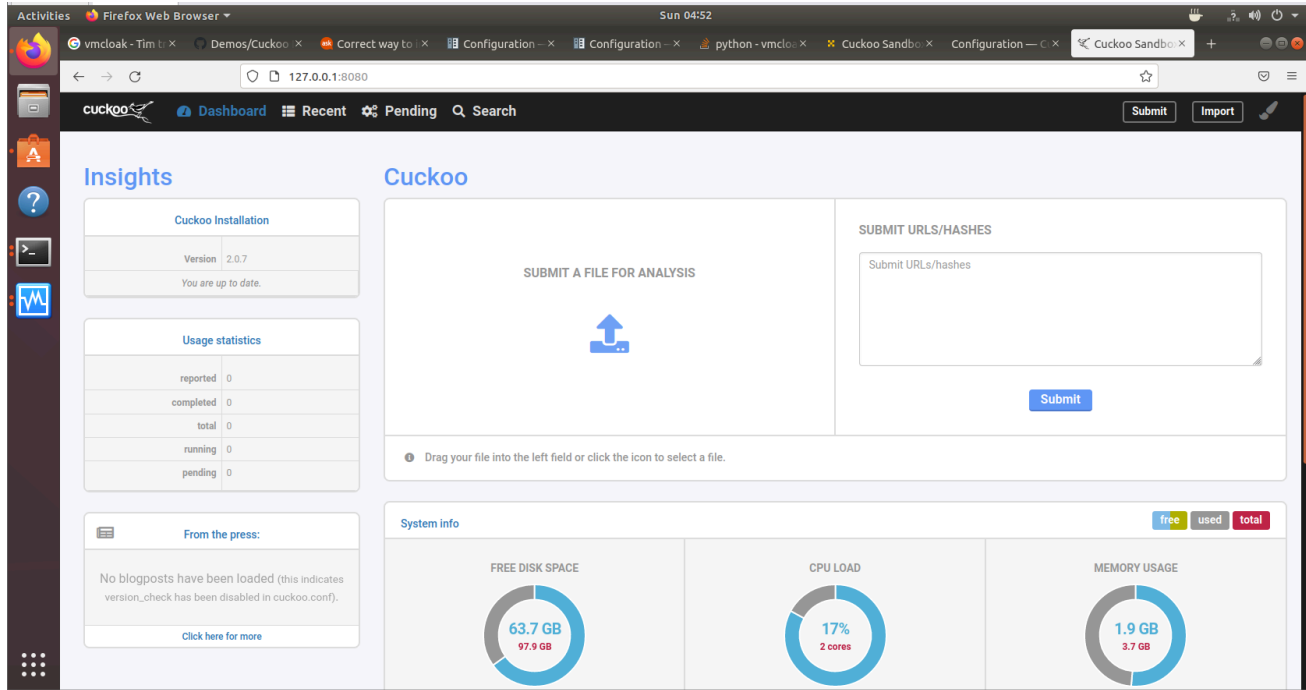
flux
evergreen
WebRTC-Experiment
react-native-firebase
ndb

```

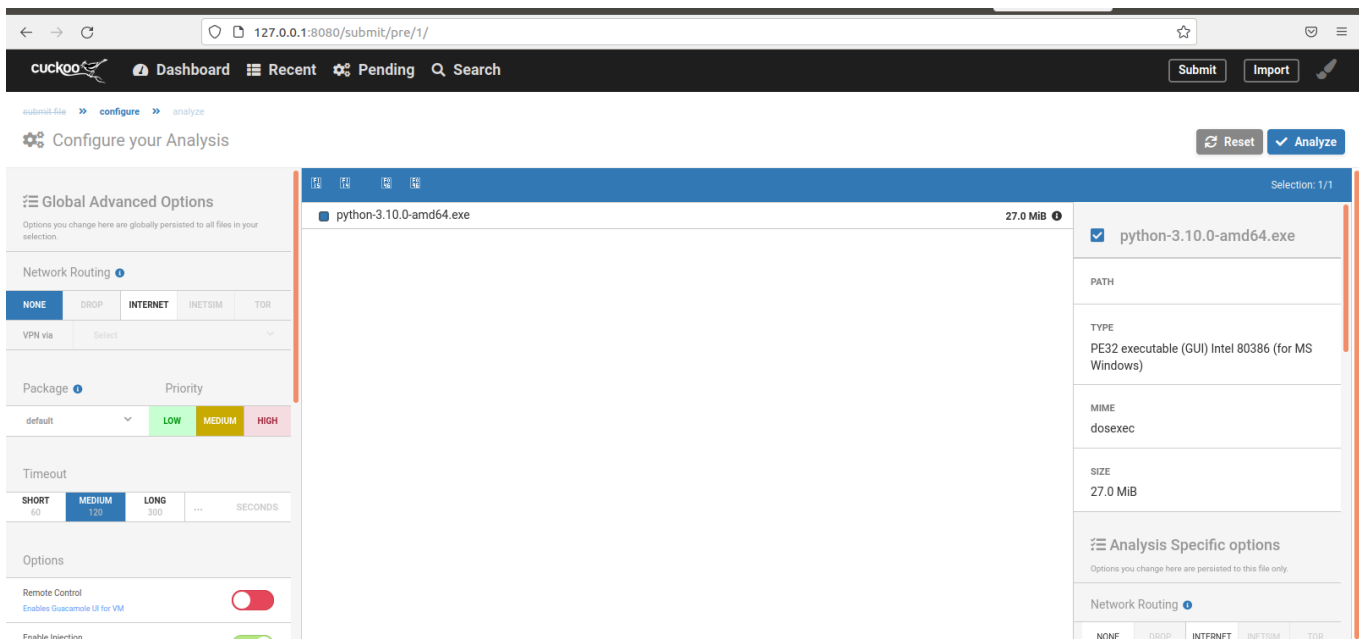
Nhấp đúp (hoặc nhấn Enter) để chỉnh sửa

3. Sinh viên tự tìm hiểu, cài đặt (<https://cuckoo.sh/docs/introduction/index.html>), thực hiện và trình bày phân tích động một tập tin PE.

- Sau khi cài đặt cuckoo, ta có 1 giao diện web như sau



- Upload file lên web GUI, ở đây ta chọn file "python-3.10.0-amd64.exe" làm ví dụ



- Để cuckoo phân tích, sau một lúc ta thu được một số kết quả như sau

Cuckoo SandboxNew Tab127.0.0.1:8080/analysis/1/summaryDashboardRecentPendingSearchSubmitImport

Summarypython-3.10.0-amd64.exe

File python-3.10.0-amd64.exe

SummaryDownloadResubmit sample

Size	27.0MB
Type	PE32 executable (GUI) Intel 80386, for MS Windows
MD5	c3917c08a7fe85db7203da6dcaa99a70
SHA1	3ee4e92a8ef94c70fb56859503fdc805d217d689
SHA256	cb580eb7dc55f9198e650f016645023e8b2224cf7d033857d12880b46c5c94ef
SHA512	Show SHA512
CRC32	1103848D
ssdeep	None
PDB Path	C:\agent_work\8\s\build\ship\vx86\burn.pdb
Yara	None matched

Score

This file shows numerous signs of malicious behavior.
The score of this file is 2.8 out of 10.

Please notice: The scoring system is currently still in development and should be considered an alpha feature.

Feedback

Expecting different results? Send us this analysis and we will inspect it. [Click here](#)

Information on Execution

Category	Started	Completed	Duration	Routine	Logs
Analysis					

Cuckoo SandboxNew Tab127.0.0.1:8080/analysis/1/summaryDashboardRecentPendingSearchSubmitImport

FILEApril 7, 2024, 5:54 a.m.April 7, 2024, 5:55 a.m.55 secondsnoneShow Analyzer LogShow Cuckoo Log

Signatures

Checks if process is being debugged by a debugger (1 event)

Time & API	Arguments	Status	Return	Repeated
IsDebuggerPresent April 6, 2024, 8:01 p.m.			0	0

This executable has a PDB path (1 event)

pdb_path	C:\agent_work\8\s\build\ship\vx86\burn.pdb			
----------	--------------------------------------------	--	--	--

The executable contains unknown PE section names indicative of a packer (could be a false positive) (1 event)

section	.wixburn			
---------	----------	--	--	--

Creates executable files on the filesystem (2 events)

file	C:\Windows\Temp\{30FDED5E-8FC5-4A3C-9971-62467BA07B81}\.ba\PythonBA.dll			
file	C:\Windows\Temp\{2C99A7B9-32CE-49A9-B7D1-B85CC28559A5}\.cr\python-3.10.0-amd64.exe			

Drops a binary and executes it (1 event)

file	C:\Windows\Temp\{2C99A7B9-32CE-49A9-B7D1-B85CC28559A5}\.cr\python-3.10.0-amd64.exe			
------	------------------------------------------------------------------------------------	--	--	--

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

Python

https://colab.research.google.com/drive/14X5r7KvAexkJLwPdYqr7dwAp0xT3c4rB?usp=sharing#scrollTo=U4gWOLIP5QsS&printMode=true

10/21

```

# B1. Import thư viện PyGithub để gọi API của Github và sử dụng module base64 để mã hoá và giải mã tập tin.
import os
from github import Github
import base64

# B2. Ta phải cung cấp thông tin chứng thực và đưa ra một truy vấn Python trong repository
username = "MN911718"
password = "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"
target_dir = "/content/drive/MyDrive/NT522.O21.ANTT_Lab2/PythonSampleCode/"
g = Github(username, password)
repositories = g.search_repositories(query="language:python")
n = 5
i = 0
# B3. Duyệt danh sách repository trả về
for repo in repositories:
    reponame = repo.name
    target_dir_of_repo = target_dir + "\\" + reponame
    print(reponame)
    try:
# B4. Tạo thư mục lưu trữ
        os.mkdir(target_dir_of_repo)
        i += 1
        contents = repo.get_contents("")
# B5. Ta thêm tất cả các thư mục của repository vào hàng đợi để liệt kê tất cả các tập tin trong thư mục
        while len(contents) > 1:
            file_content = contents.pop(0)
            if file_content.type == "dir":
                contents.extend(repo.get_contents(file_content.path))
            else:
# B6. Nếu kiểm tra một tập tin không phải là thư mục thì kiểm tra phần mở rộng có phải là .py
                st = str(file_content)
                filename = st.split('')[1].split('')[0]
                extension = filename.split(".")[1]
                if extension == "py":
# B7. Nếu là .py thì sẽ ghi ra tập tin
                    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

youtube-dl
ansible

```

Powershell

```

# B1. Import thư viện PyGitHub để gọi API của Github và sử dụng module base64 để mã hoá và giải mã tập tin.
import os
from github import Github
import base64

# B2. Ta phải cung cấp thông tin chứng thực và đưa ra một truy vấn Powershell trong repository
username = "MN911718"
password = "ghp_li0deOKT6bWco1JASAKoYufq0ILhDR07wRMa"
target_dir = "/content/drive/MyDrive/NT522.O21.ANTT_Lab2/PowershellSampleCode/"
g = Github(username, password)
repositories = g.search_repositories(query="language:powershell")
n = 5
i = 0
# B3. Duyệt danh sách repository trả về
for repo in repositories:
    reponame = repo.name
    target_dir_of_repo = target_dir + "\\" + reponame
    print(reponame)
    try:
# B4. Tạo thư mục lưu trữ
        os.mkdir(target_dir_of_repo)
        i += 1
        contents = repo.get_contents("")
# B5. Ta thêm tất cả các thư mục của repository vào hàng đợi để liệt kê tất cả các tập tin trong thư mục
        while len(contents) > 1:
            file_content = contents.pop(0)
            if file_content.type == "dir":
                contents.extend(repo.get_contents(file_content.path))
            else:
# B6. Nếu kiểm tra một tập tin không phải là thư mục thì kiểm tra phần mở rộng có phải là .ps1
                st = str(file_content)
                filename = st.split('')[1].split('')[0]
                extension = filename.split(".")[1]
                if extension == "ps1":
# B7. Nếu là .ps1 thì sẽ ghi ra tập tin
                    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

        core
        Scoop
        Windows10Debloater
        WSL
        PowerSploit

```

› g) Đo lường sự giống nhau giữa hai chuỗi

[] 3 ô bị ẩn

h) Đo lường mức độ giống nhau giữa hai tập tin

f) Phân loại tập tin theo kiểu

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

```

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/NT522.O21.ANTT_Lab2/JavaScripts/"
python_path = "/content/drive/MyDrive/NT522.O21.ANTT_Lab2/PythonSampleCode/"
powershell_path = "/content/drive/MyDrive/NT522.O21.ANTT_Lab2/PowershellSampleCode/"

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

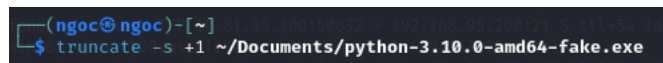
text_clf.fit(X_train, y_train)
y_test_pred =text_clf.predict(X_test)
print(accuracy_score(y_test,y_test_pred))
print(confusion_matrix(y_test, y_test_pred))

0.8125
[[4 1 0]
 [0 5 2]
 [0 0 4]]

```

✓ 6. Sinh viên cho biết quả của đoạn code sau

B1. Đầu tiên 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 byte



```

ngoc@ngoc:~$ truncate -s +1 ~/Documents/python-3.10.0-amd64-fake.exe

```

B2. Dùng hexdump để xem sự khác nhau giữa hai tập tin trước và sau

Ta thấy có sự khác biệt ở byte cuối cùng.

```
(ngoc@ngoc)-[~/Documents]
$ 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

(ngoc@ngoc)-[~/Documents]
$ 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
```

B3. Dùng ssdeep để so sánh 2 tập tin

```
import ssdeep
hash1 = ssdeep.hash_from_file("/content/drive/MyDrive/NT522.O21.ANTT_Lab2/python-3.10.0-amd64.exe")
hash2 = ssdeep.hash_from_file("/content/drive/MyDrive/NT522.O21.ANTT_Lab2/python-3.10.0-amd64-fake.exe")
ssdeep.compare(hash1, hash2)
```

```
100
```

Từ kết quả bên trên ta có thể thấy được với 2 file chỉ khác nhau 1 byte, ssdeep không phát hiện được.

ssdeep là một công cụ dùng để tạo và so sánh các bản tóm tắt của tệp, nó tạo ra bản tóm tắt dựa trên nội dung của tệp và sắp xếp các khối dữ liệu thành các "chunk". Tuy nhiên, nó không chỉ dựa trên từng byte đơn lẻ mà còn trên cấu trúc và phân phối của dữ liệu trong tệp. Vì vậy, một byte thay đổi có thể không ảnh hưởng đến cấu trúc tổng thể của dữ liệu, do đó ssdeep có thể không phát hiện được sự thay đổi nhỏ này.

✓ i) Trích xuất N-grams

```
!pip install nltk
```

```
# B1. Import thư viện collections để đếm và ngrams từ nltk để trích xuất N-grams
```

```
import collections
from nltk import ngrams
```

```
# B2. Chọn tập tin phân tích
```

```
file_to_analyze = "/content/drive/MyDrive/NT522.O21.ANTT_Lab2/python-3.10.0-amd64.exe"
```

```
# B3. Định nghĩa hàm đọc tập tin từ bytes
```

```
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
```

```
# B4. Viết một hàm lấy một chuỗi bytes thành N-grams
```

```
def byte_sequence_to_Ngrams(byte_sequence, N):
    """Creates a list of N-grams from a byte sequence."""
    Ngrams = ngrams(byte_sequence, N)
    return list(Ngrams)
```

```
# B5. Viết một hàm đọc một tập tin và lấy được N-grams của nó
```

```
def binary_file_to_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)
```

```
# B6. Định nghĩa với N-4 thu được số lượng 4-grams
```

```
extracted_Ngrams = binary_file_to_Ngram_counts(file_to_analyze, 4)
```

```
# B7. Hiển thị 10 kết quả đầu 4-grams phổ biến trong tập tin
```

```
for item in extracted_Ngrams.most_common(10):
    print(item)
```

```
((0, 0, 0, 0), 24290)
((139, 240, 133, 246), 1920)
((32, 116, 111, 32), 1791)
((255, 255, 255, 255), 1671)
((108, 101, 100, 32), 1522)
((100, 32, 116, 111), 1519)
((97, 105, 108, 101), 1513)
((105, 108, 101, 100), 1513)
((70, 97, 105, 108), 1505)
((101, 100, 32, 116), 1503)
```

✓ j) Chọn N-grams tốt nhất

B1. Chọn thư viện, xác định N và định nghĩa đường dẫn.

```
from os import listdir
from os.path import isfile, join
directories = ["/content/drive/MyDrive/NT522.O21.ANTT_Lab2/Benign PE Samples", "/content/drive/MyDrive/NT522.O21.ANTT_Lab2/Malicious PE Samp
N = 2
```

B2. Đếm tất cả N-grams trong tập tin

```
Ngram_counts_all_files = collections.Counter([])
for dataset_path in directories:
    all_samples = [f for f in listdir(dataset_path) if
isfile(join(dataset_path, f))]
    for sample in all_samples:
        file_path = join(dataset_path, sample)
        Ngram_counts_all_files += binary_file_to_Ngram_counts(file_path, N)
```

B3. Ta sẽ thêm với K=1000 với N-grams thường gặp

```
K1 = 1000
K1_most_frequent_Ngrams = Ngram_counts_all_files.most_common(K1)
K1_most_frequent_Ngrams_list = [x[0] for x in K1_most_frequent_Ngrams]
```

B4. Phương thức featurize_sample sẽ sử dụng mẫu và xuất số lần xuất hiện của

N-grams phổ biến trong chuỗi bytes của nó.

```
def featurize_sample(sample, K1_most_frequent_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 = binary_file_to_Ngram_counts(sample, N)
    for i in range(K1):
        feature_vector[i] = file_Ngrams[K1_most_frequent_Ngrams_list[i]]
    return feature_vector
```

B5. Dùng hàm featurize_sample duyệt qua tất cả tập tin và gán nhãn cho chúng.

```
directories_with_labels = [("/content/drive/MyDrive/NT522.O21.ANTT_Lab2/Benign PE Samples", 0),
("/content/drive/MyDrive/NT522.O21.ANTT_Lab2/Malicious PE Samples", 1)]
X = []
y = []
for dataset_path, label in directories_with_labels:
    all_samples = [f for f in listdir(dataset_path) if isfile(join(dataset_path, f))]
    for sample in all_samples:
        file_path = join(dataset_path, sample)
        X.append(featurize_sample(file_path, K1_most_frequent_Ngrams_list))
        y.append(label)
```

B6. Import thư viện để chọn thuộc tính và số lượng thuộc tính muốn lấy

```
from sklearn.feature_selection import SelectKBest, mutual_info_classif, chi2
K2 = 10
```

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

- **Frequency:** Chọn N-grams phổ biến

```
import numpy as np
```

```
X = np.asarray(X)
X_top_K2_freq = X[:, :K2]
print(X_top_K2_freq)
```

```
[[ 802224 448826 22478 17957 7827 11493 6227 25273 42
 7087]
 [ 20146 4014 13 1342 60 0 3 1062 2495
 9]
 [ 21401 1040 4948 3695 2918 0 1876 352 1554
 1304]
 [ 14677 1493 1949 1636 920 1459 679 308 18
 799]
 [ 35697 6257 6263 7671 2239 6059 1337 1591 28
 2724]
 [ 4160 494 0 99 11 0 0 115 137
 0]
 [ 11257 827 1223 463 681 0 507 181 440
 456]
 [ 7719 274 628 384 376 0 185 61 458
 274]
 [2028238 631237 216934 118021 103955 166298 111696 82528 87
 73376]
 [ 4216 114 0 57 6 0 0 87 137
 0]
 [ 8 1 0 1 1 0 2 2 1
 1]
 [ 9 1 1 5 2 1 1 2 2
 3]
 [ 3628 175 0 94 2 0 0 69 84
 0]
 [ 30694 4419 2415 4993 2102 9 1124 1894 1807
 674]
 [ 5507 471 1 121 12 0 1 69 97
 0]
 [ 8 3 0 4 0 1 0 1 0
 0]
 [ 6 4 0 1 0 3 0 0 0
 0]
 [ 44667 6048 9250 5880 4710 1 2915 1507 3073
 2474]
 [ 9 0 0 1 1 1 0 0 0
 0]
 [ 3119 78 0 56 2 0 0 17 100
 0]
 [ 12 11 1 5 0 3 1 3 1
 1]
 [ 9 1 0 2 1 0 0 0 0
 0]
 [ 3096 47 0 37 2 0 0 19 108
 0]
 [ 7 3 0 5 0 0 0 0 0
 0]
 [ 2481 1 0 133 9 1 1 3 0
 0]
 [ 110759 130086 10 1925 72 6 2 5472 1490
 5]
 [ 9410 4453 1 318 37 2 0 421 481
 1]
 [ 52163 8162 15778 6626 7701 0 5849 2012 5038
 4520]
 [ 65038 11212 15 2845 94 10 5 1891 2811
 57]
```

• **Mutual information:** Chọn N-grams có xếp hạng cao theo thuật toán mutual information

```
mi_selector = SelectKBest(mutual_info_classif, k=K2)
X_top_K2_mi = mi_selector.fit_transform(X, y)
print(X_top_K2_mi)
```

```
[[ 12 168 121 152 120 24 234 173 54 180]
 [ 2 0 0 22 0 58 5 33 3 0]
 [ 2 20 35 18 63 43 10 24 17 0]
 [ 1 10 14 9 11 7 20 15 6 1]
 [ 5 36 42 63 15 10 46 25 10 3]
 [ 0 0 0 6 0 9 11 2 0 0]
 [ 4 9 9 10 18 12 14 4 6 0]
 [ 0 3 2 0 10 5 2 1 7 0]
 [1461 1545 1229 1049 958 410 1805 900 646 320]
 [ 0 0 0 0 0 1 0 3 0 0]
 [ 0 0 2 0 1 1 1 0 1 1]
```



```
[ 2 2 0 2 0 2 0 1 0 0]
[ 0 0 0 0 1 0 0 0 0 0]
[ 12 18 21 3 29 43 23 28 25 8]
[ 0 0 0 0 0 0 5 24 0 0]
[ 0 0 0 0 0 0 0 0 0 0]
[ 3 0 1 0 0 0 0 1 1 1]
[ 15 75 95 211 149 62 9 47 87 2]
[ 0 0 0 0 1 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0]
[ 2 2 1 1 3 2 1 0 1 5]
[ 0 0 0 1 1 0 1 0 0 1]
[ 0 0 0 0 0 1 0 0 0 0]
[ 2 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 2 8 0 1]
[ 8 2 1 5 0 19 22 30 2 61]
[ 2 1 0 7 0 11 5 8 0 0]
[ 6 113 210 129 218 179 3 93 227 5]
[ 61 4 0 54 0 100 37 90 9 143]
[ 27 3 0 20 0 38 22 67 5 45]
[ 492 2 5 35 0 35 13 34 1 1]
[ 12 2 2 5 0 14 12 30 1 30]
[ 12 1 0 1 0 5 8 11 0 30]
[ 29 2 0 1 0 39 18 46 2 29]
[ 3 19 11 6 105 25 23 30 40 0]
[ 5 2 0 54 2 39 8 79 4 0]
[ 10 1 0 9 2 17 5 22 2 0]
[ 57 2 6 19 0 45 68 18 4 154]
[ 7 0 2 2 1 2 5 4 1 8]
[ 0 1 0 1 0 0 0 0 0 0]
[ 162 8 3 5 4 23 15 51 2 736]
[ 0 1 0 3 1 0 0 0 2 0]
[ 38 2 2 4 2 13 2 17 1 0]
[ 0 0 0 0 1 1 0 1 0 0]
[ 0 3 0 11 2 28 2 7 1 0]
[ 18 552 515 217 396 488 46 319 157 55]
[ 12 2 0 34 8 110 14 40 5 10]
[ 81 4 1 0 3 3 7 8 1 368]
[ 16 11 7 37 5 99 36 105 9 205]
[ 1 1 0 0 1 0 1 1 1 1]
[ 0 0 4 2 14 1 19 2 3 0]
[ 0 0 0 1 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0]
[ 1 4 8 2 2 2 11 3 10 1]
[ 16 76 239 122 228 225 24 53 933 52]
[ 0 0 1 0 2 2 0 0 0 0]
[ 0 4 16 0 8 16 0 4 6 0]
[ 0 2 2 0 6 3 0 1 4 0]
```

• **Chi-squared:** Chọn N-grams có xếp hạng cao theo thuật toán chi squared

```
chi2_selector = SelectKBest(chi2, k=K2)
X_top_K2_ch2 = chi2_selector.fit_transform(X, y)
print(X_top_K2_ch2)
```

```
[[ 1971 436 367 148 143 376 4 13 0 0]
[ 68 523 248 124 81 111 20 4 33 0]
[ 391 98 130 31 55 32 6 6 0 0]
[ 220 23 33 4 11 16 2 1 0 0]
[ 722 35 86 39 16 8 4 1 1 0]
[ 66 77 61 10 12 31 20 0 3 0]
[ 137 19 26 2 15 2 3 2 0 0]
[ 90 6 12 1 7 2 1 1 0 0]
[ 21020 3204 3036 1540 1109 1075 59 71 36 1]
[ 7 53 17 8 8 4 2 2 3 0]
[ 0 0 1 3 0 1 1 0 1 2]
[ 2 1 2 5 2 1 1 4 0 2]
[ 9 19 13 2 0 5 4 2 0 0]
[ 217 44 148 10 19 23 2 1 0 1]
[ 88 36 84 103 10 25 31 1 3 0]
[ 1 0 0 0 0 2 0 0 0 1]
[ 1 1 0 0 1 0 1 1 2 1]
[ 699 173 255 75 96 33 4 11 0 0]
[ 0 0 0 0 1 0 0 0 0 0]
[ 6 16 11 3 0 3 4 0 0 0]
[ 2 2 0 0 1 1 1 1 4 3]
[ 0 1 0 0 2 0 0 0 0 0]
[ 5 15 11 2 0 3 2 0 0 0]
[ 1 0 0 0 1 1 0 0 0 1]
[ 0 0 6 0 0 1 1 0 0 0]
[ 104 193 241 87 52 66 33 3 24 1]
[ 81 76 140 34 18 31 43 1 8 0]
[ 1072 266 430 68 142 50 4 18 0 0]
[ 356 587 734 396 149 300 103 11 111 0]
```

```
[ 205 392 312 300 110 135 74 30 32 2]
[ 224 457 422 265 70 108 90 3 47 0]
[ 75 147 128 65 32 51 31 3 26 0]
[ 40 75 75 29 9 25 14 0 12 0]
[ 110 471 284 126 77 98 22 3 28 0]
[ 340 127 95 13 38 8 2 2 0 0]
[ 328 776 626 485 188 257 84 16 59 0]
[ 79 221 165 98 43 47 26 4 19 0]
[ 206 284 423 171 43 114 97 5 34 0]
[ 9 26 26 7 3 14 1 0 1 1]
[ 0 0 0 0 1 0 0 0 0 0]
[ 75 192 165 72 57 54 19 3 27 2]
[ 0 0 1 0 1 0 0 0 0 0]
[ 72 127 126 48 30 81 27 5 14 1]
[ 3 1 1 3 1 1 0 0 2 1]
[ 149 222 147 90 54 121 40 3 24 0]
[14069 916 961 522 441 400 41 84 48 6]
[ 377 489 543 129 112 179 53 1 17 0]
[ 16 22 42 9 5 13 5 1 7 1]
[ 164 1087 719 319 217 174 32 256 77 4]
[ 1 1 1 2 2 0 0 0 2 1]
[ 41 13 14 4 1 17 1 0 0 0]
[ 0 0 1 0 1 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0]
[ 105 14 11 3 3 2 1 0 0 0]
[ 1038 376 391 134 189 113 9 24 3 1]
[ 15 2 2 0 0 0 1 0 0 0]
[ 92 22 19 2 5 0 1 0 0 0]
[ 30 2 11 0 0 4 0 0 0 0]
```

4. Xây dựng trình phát hiện phần mềm độc hại bằng phân tích tĩnh

Trình phân tích này sử dụng cả 2 bộ thuộc tính trích xuất từ PE header và N-grams. Sử dụng tập dữ liệu Benign PE Samples và Malicious PE Sample.

- B1. Tạo list các mẫu và gán nhãn cho chúng.
- B2. Chia dữ liệu train-test
- B3. Các hàm lấy thuộc tính
- B4. Chọn 100 thuộc tính phổ biến với 2-grams
- B5. Trích xuất số lượng N-grams count, section names, imports và số lượng sections của mỗi mẫu trong train-test.
- B6. Sử dụng hàm băm tfidf để chuyển imports, section names từ văn bản thành dạng số
- B7. Kết hợp các vector thuộc tính thành 1 mảng.
- B8. Ta huấn luyện bằng phân loại Random Forest cho tập train
- B9. Thu thập các thuộc tính của tập test, giống như tập huấn luyện
- B10. Ta chuyển đổi vector từ thuộc tính test, và kiểm tra kết quả của trình phân loại.

✓ 8. Sinh viên hoàn thành các bước trên

```
!pip install collection
!pip install ngrams
```

```
Requirement already satisfied: collection in /usr/local/lib/python3.10/dist-packages (0.1.6)
Collecting ngrams
  Downloading ngrams-1.0.3.tar.gz (1.3 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: ngrams
  Building wheel for ngrams (setup.py) ... done
  Created wheel for ngrams: filename=ngrams-1.0.3-py3-none-any.whl size=1557 sha256=e51a6ef6cde85ce4f393b19e2ef18c69031cd146a9b9b1e84c4e
  Stored in directory: /root/.cache/pip/wheels/b9/cd/04/2d6e1de5eb5a26b2c272d258871ea4449cad1092abf3e1b99b
Successfully built ngrams
Installing collected packages: ngrams
Successfully installed ngrams-1.0.3
```

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

```
import os
import pefile
import pandas as pd
import collections
from os import listdir
from os.path import isfile, join
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
from nltk import ngrams

Benign_path = "/content/drive/MyDrive/NT522.O21.ANTT_Lab2/Benign PE Samples"
Malicious_path = "/content/drive/MyDrive/NT522.O21.ANTT_Lab2/Malicious PE Samples"

# Đọc tất cả các kiểu tập tin. Tạo một mảng nhãn có giá trị 0, 1 đại diện cho Benign và Malicious file
corpus = []
labels = []
file_types_and_labels = [(Benign_path, 0), (Malicious_path, 1)]

for files_path, label in file_types_and_labels:
    files = os.listdir(files_path)
    for file_name in files:
        file_path = os.path.join(files_path, file_name)
        corpus.append(file_path)
        labels.append(label)
```

B2. Chia dữ liệu train-test

```
X_train, X_test, y_train, y_test = train_test_split(corpus, labels, test_size=0.3, random_state=11)
```

B3. Các hàm lấy thuộc tính

```
# Định nghĩa hàm đọc tập tin từ bytes
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

# hàm lấy một chuỗi bytes thành N-grams
def byte_sequence_to_Ngrams(byte_sequence, N):
    """Creates a list of N-grams from a byte sequence."""
    Ngrams = ngrams(byte_sequence, N)
    return list(Ngrams)

# hàm đọc một tập tin và lấy được N-grams của nó
def binary_file_to_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_frequent_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 = binary_file_to_Ngram_counts(sample, N)
    for i in range(K1):
        feature_vector[i] = file_Ngrams[K1_most_frequent_Ngrams_list[i]]
    return feature_vector

#Định nghĩa phương thức thu thập tên của sections và chuẩn hoá chúng.
def get_section_names(pe):
    list_of_section_names = []
    for sec in pe.sections:
        try:
            normalized_name = sec.Name.decode().replace("\x00", "").lower()
        except UnicodeDecodeError:
            # Handle decoding error by replacing problematic bytes or ignoring them
            normalized_name = sec.Name.decode('latin-1').replace("\x00", "").lower()
        list_of_section_names.append(normalized_name)
    return list_of_section_names

#Định nghĩa một phương thức thuận tiện trong việc tiền xử lý import
def preprocess_imports(list_of_DLLs):
    return [x.decode().split(".")[0].lower() for x in list_of_DLLs]

#Định nghĩa hàm thu thập import từ tập tin
def get_imports(pe):
    list_of_imports = []
    for entry in pe.DIRECTORY_ENTRY_IMPORT:
        list_of_imports.append(entry.dll)
    return preprocess_imports(list_of_imports)
```

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

```
# Đếm tất cả N-grams trong tập tin
Ngram_counts_all_files = collections.Counter([])
N = 2
for file_path in corpus:
    Ngram_counts_all_files += binary_file_to_Ngram_counts(file_path, N)

# Chọn ra K1 Ngrams phổ biến nhất
K1 = 100
K1_most_frequent_Ngrams = Ngram_counts_all_files.most_common(K1)
K1_most_frequent_Ngrams_list = [x[0] for x in K1_most_frequent_Ngrams]
```

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

```
# N-grams count
Ngrams_cnt_train = []

for file_path in X_train:
    Ngrams_cnt_train.append(featurize_sample(file_path, K1_most_frequent_Ngrams_list))

# section names, imports và số lượng sections
section_names_train = []
imports_train = []
num_sections_train = []
list_drop = []
i=1
for file_path in X_train:
    try:
        pe = pefile.PE(file_path)
        imports = get_imports(pe)
        n_sections = len(pe.sections)
        sec_names = get_section_names(pe)

        imports_train.append(imports)
        num_sections_train.append(n_sections)
        section_names_train.append(sec_names)
    except Exception as e:
        print(i, file_path, e)
        list_drop.append(file_path)
```