

BÁO CÁO THỰC HÀNH

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

Lab 2: Set up your machine learning for Cybersecurity Arsenal

GVHD: Nguyễn Hữu Quyền

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT522.021.ATCL.2

Nhóm: 7

STT	Họ và tên	MSSV	Email
1	Hồ Ngọc Thiện	21522620	21522620@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:¹

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

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

¹ Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

1. Phân tích mã độc

a) Tính toán hàm băm của một mẫu

⑧ **Bài tập (yêu cầu làm)**

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

Kết quả từ thư viện hashlib của python:

```

✓ 0s [1] import sys
    import hashlib
    filename = "python-3.10.0-amd64.exe"

✓ 0s [2] BUF_SIZE = 65536
        md5 = hashlib.md5()
        sha256 = hashlib.sha256()

✓ 0s [3] ➜ with open(filename, "rb") as f:
            while True:
                data = f.read(BUF_SIZE)
                if not data:
                    break
                md5.update(data)
                sha256.update(data)

✓ 0s [4] print("MD5: {}".format(md5.hexdigest()))
print("SHA256: {}".format(sha256.hexdigest()))

➜ MD5: c3917c08a7fe85db7203da6dcaa99a70
SHA256: cb580eb7dc55f9198e650f016645023e8b2224cf7d033857d12880b46c5c94ef

```

VirusTotal:

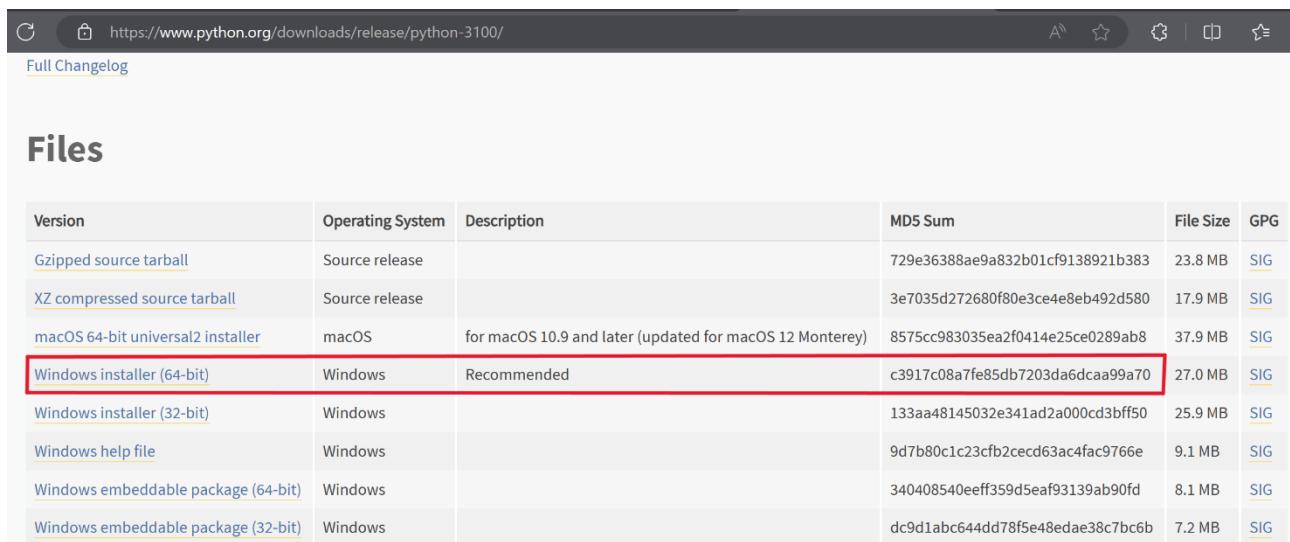
The screenshot shows the VirusTotal analysis page for the file hash cb580eb7dc55f9198e650f016645023e8b2224cf7d033857d12880b46c5c94ef. The page has a dark theme with a navigation bar at the top. Below the navigation, there are tabs for DETECTION, DETAILS (which is selected), RELATIONS, BEHAVIOR, TELEMETRY, and COMMUNITY (with 12 items). A banner at the bottom of the main content area encourages users to join the community. The 'Basic properties' section lists the following details:

MD5	c3917c08a7fe85db7203da6dcaa99a70
SHA-1	3ee4e92a8ef94c70fb56859503fdc805d217d689
SHA-256	cb580eb7dc55f9198e650f016645023e8b2224cf7d033857d12880b46c5c94ef

Ta thấy 2 hàm băm md5 và sha256 hoàn toàn trùng khớp

Website Python:

Ta có mã băm md5 trên trang web của python cũng trùng khớp với 2 phương pháp trên



Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		729e36388ae9a832b01cf9138921b383	23.8 MB	SIG
XZ compressed source tarball	Source release		3e7035d272680f80e3ce4e8eb492d580	17.9 MB	SIG
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later (updated for macOS 12 Monterey)	8575cc983035ea2f0414e25ce0289ab8	37.9 MB	SIG
Windows installer (64-bit)	Windows	Recommended	c3917c08a7fe85db7203da6dcaa99a70	27.0 MB	SIG
Windows installer (32-bit)	Windows		133aa48145032e341ad2a000cd3bfff50	25.9 MB	SIG
Windows help file	Windows		9d7b80c1c23cfb2cecd63ac4fac9766e	9.1 MB	SIG
Windows embeddable package (64-bit)	Windows		340408540eff359d5eaf93139ab90fd	8.1 MB	SIG
Windows embeddable package (32-bit)	Windows		dc9d1abc644dd78f5e48edae38c7bc6b	7.2 MB	SIG

b) YARA

```
dummy_rule2 .\rules.yara
PS C:\Users\Ngoc Thien\Desktop> cat .\rules.yara
rule is_a_pdf
{
    strings:
        $pdf_magic = {25 50 44 46}
    condition:
        $pdf_magic at 0
}
rule dummy_rule1
{
    condition:
        false
}
rule dummy_rule2
{
    condition:
        true
}
PS C:\Users\Ngoc Thien\Desktop> yara .\rules.yara .\NT330.021.ATCL.1-Lab2-Nhom07.pdf
is_a_pdf .\NT330.021.ATCL.1-Lab2-Nhom07.pdf
dummy_rule2 .\NT330.021.ATCL.1-Lab2-Nhom07.pdf
```

c) Kiểm tra PE header

Lab 1: Memory Forensic

4

```
✓ [108] pip install pefile
Requirement already satisfied: pefile in /usr/local/lib/python3.10/dist-packages (2023.2.7)

✓ [110] import pefile
desired_file = "drive/MyDrive/Lab2/python-3.10.0-amd64.exe"
pe = pefile.PE(desired_file)

✓ [111] 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'

✓ [112] for section in pe.sections:
    print(
        section.Name,
        hex(section.VirtualAddress),
        hex(section.Misc_VirtualSize),
        section.SizeOfRawData,
    )

    b'.text\x00\x00\x00' 0x1000 0x49937 301568
    b'.rdata\x00\x00' 0x4b000 0x1ed60 126464
    b'.data\x00\x00\x00' 0x6a000 0x1730 2560
    b'.wixburn' 0x6c000 0x38 512
    b'.rsrc\x00\x00\x00' 0x6d000 0x165fc 91648
    b'.reloc\x00\x00' 0x84000 0x3dfc 15872

✓ [113] print(pe.dump_info())
-----DOS_HEADER-----
[IMAGE_DOS_HEADER]
0x0      0x0    e_magic:          0x5A4D
0x2      0x2    e_cblp:          0x90
0x4      0x4    e_cp:            0x3
0x6      0x6    e_crlc:          0x0
0x8      0x8    e_cparhdr:       0x4
0xA      0xA    e_minalloc:      0x0
0xC      0xC    e_maxalloc:      0xFFFF
0xE      0xE    e_ss:             0x0
0x10     0x10   e_sp:            0xB8
0x12     0x12   e_csum:          0x0
0x14     0x14   e_ip:            0x0
0x16     0x16   e_cs:            0x0
0x18     0x18   e_lfarlc:        0x40
0x1A     0x1A   e_ovno:          0x0
```

⑧ Bài tập (yêu cầu làm)

2. Sinh viên cho biết quả của đoạn code trên

```

✓ 21s [1] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

✓ 8s [3] pip install pefile
Collecting pefile
  Downloading pefile-2023.2.7-py3-none-any.whl (71 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.8/71.8 kB 1.3 MB/s eta 0:00:00
Installing collected packages: pefile
Successfully installed pefile-2023.2.7

[4] import pefile
from os import listdir
from os.path import isfile, join
directories = ["drive/MyDrive/Lab2/Benign PE Samples", "drive/MyDrive/Lab2/Malicious PE Samples"]

[5] 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

[6] 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]

[7] 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)

[11] 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
    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)
    except Exception as e:
        print(e)
        print("Unable to obtain imports from " + file_path)

0s [ ] print("Imports: %s" % imports_corpus)
print("The number of sections: %s" % num_sections)
print("Sections: %s" % section_names)

[ ] Imports: [['api-ms-win-crt-runtime-l1-1-0', 'api-ms-win-crt-string-l1-1-0', 'api-ms-win-crt-private-l1-1-0', 'api-ms-win-core-libraryloader-l1-2-0', 'api-ms-win-core']
The number of sections: [6, 5]
Sections: [['.text', '.rdata', '.pdata', '.didat', '.reloc'], ['.text', '.data', '.rdata', '.bss', '.idata']]

```

Lab 1: Memory Forensic

④ Bài tập (yêu cầu làm)

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.

Quá trình cài đặt cuckoo được tham khảo và làm theo tài liệu sau:

<https://github.com/ForeGuards/Cuckoo-Installation-Guide/blob/main/installation.txt>

-Sau khi cài đặt xong tiến hành phân tích trên cuckoo.

Kích hoạt môi trường ảo trước khi thao tác các tác vụ:

-Thực hiện bật bộ định tuyến để có thể kết nối với máy ảo window

```
cuckoo@ubuntu:~$ workon cuckoo-test
(cuckoo-test) cuckoo@ubuntu:~$ cuckoo rooter --sudo --group cuckoo
/home/cuckoo/.virtualenvs/cuckoo-test/local/lib/python2.7/site-packages/sflock/decode/office.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
    from cryptography.hazmat.backends import default_backend
[sudo] password for cuckoo:
/home/cuckoo/.virtualenvs/cuckoo-test/local/lib/python2.7/site-packages/sflock/decode/office.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
    from cryptography.hazmat.backends import default_backend
```

-Khi thành công sẽ có dòng thông báo:

```
[root@cuckoo ~]# ./cuckoo
[cuckoo] INFO: Starting Cuckoo Rooter (group=cuckoo)!
```

-Khởi động cuckoo ở terminal khác

```
t is now deprecated in cryptography, and will be removed in the next release.
from cryptography.hazmat.backends import default_backend



Cuckoo Sandbox 2.0.7
www.cuckoosandbox.org
Copyright (c) 2010-2018

Checking for updates...
```

```
[cuckoo.core.scheduler] INFO: Using "virtualbox" as machine manager
[cuckoo.core.scheduler] INFO: Loaded 1 machine/s
[cuckoo.core.scheduler] INFO: Waiting for analysis tasks.
```

-Tiếp theo kiểm tra ip của máy

Lab 1: Memory Forensic



```
cuckoo@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.242.139 netmask 255.255.255.0 broadcast 192.168.242.255
        inet fe80::a2a4:b4a9:5b1f:cc90 prefixlen 64 scopeid 0x20<link>
          ether 00:0c:29:7b:da:82 txqueuelen 1000 (Ethernet)
            RX packets 3612722 bytes 5348077579 (5.3 GB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 678395 bytes 45448207 (45.4 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
      loop txqueuelen 1000 (Local Loopback)
        RX packets 10421 bytes 9262264 (9.2 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 10421 bytes 9262264 (9.2 MB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vboxnet0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 192.168.56.1 netmask 255.255.255.0 broadcast 192.168.56.255
      inet6 fe80::800:27ff:fe00:0 prefixlen 64 scopeid 0x20<link>
```

Khởi động môi trường ảo cuckoo và khởi động giao diện web cuckoo

```
cuckoo@ubuntu:~$ workon cuckoo-test
(cuckoo-test) cuckoo@ubuntu:~$ cuckoo web --host 192.168.242.139 --port 8080
/home/cuckoo/.virtualenvs/cuckoo-test/local/lib/python2.7/site-packages/sflock/decode/office.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography.hazmat.backends import default_backend
/home/cuckoo/.virtualenvs/cuckoo-test/local/lib/python2.7/site-packages/sflock/decode/office.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography.hazmat.backends import default_backend
Performing system checks...

System check identified no issues (0 silenced).
August 23, 2023 - 19:14:43
Django version 1.8.4, using settings 'cuckoo.web.web.settings'
Starting development server at http://192.168.242.139:8080/
Quit the server with CONTROL-C.
```

-Sau khi nhập lệnh xong sẽ có đường dẫn đến trang web cuckoo

-Giao diện web như hình bên dưới

Lab 1: Memory Forensic



The screenshot shows the Cuckoo Sandbox dashboard. On the left, there's an 'Insights' sidebar with 'Cuckoo Installation' (Version 2.0.7, up to date), 'Usage statistics' (reported 0, completed 0, total 0, running 0, pending 0), and a news section about Cuckoo Sandbox 2.0.7. The main area has a 'SUBMIT A FILE FOR ANALYSIS' section with a file upload button and a note to drag files or click the icon. To the right is a 'SUBMIT URLs/HASHES' section with a 'Submit' button. Below these are three circular gauges: 'FREE DISK SPACE' (26.0 GB / 38.8 GB), 'CPU LOAD' (11% / 4 cores), and 'MEMORY USAGE' (2.1 GB / 3.7 GB). The bottom part of the dashboard shows a list of recent files: 'Adware (004f7c2e1).zip' (779.8 kB, 18:52) and 'google-chrome-stable_current_amd64.deb' (96.7 MB, 03:24).

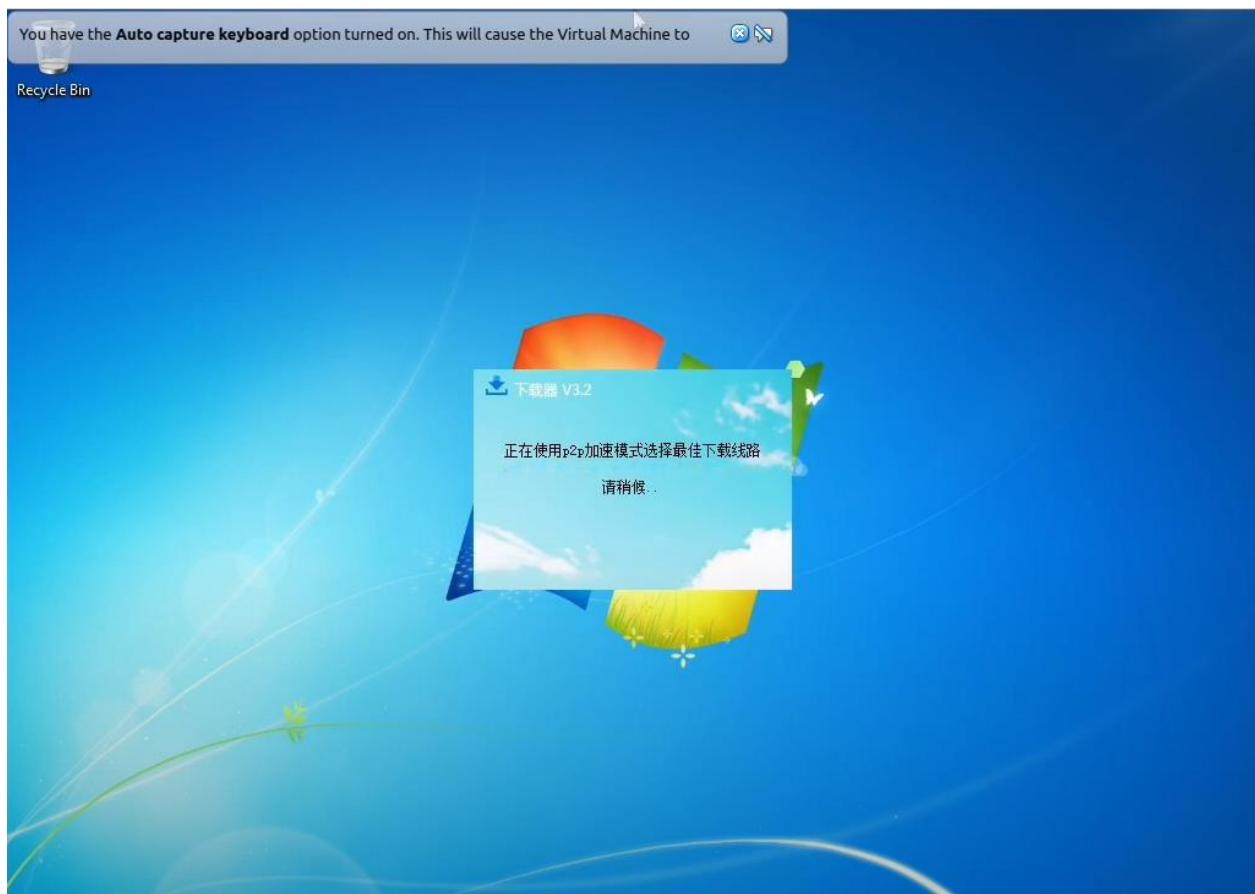
-Bỏ file cần phân tích vào

This screenshot shows a file selection dialog titled 'Open Files'. The left sidebar lists locations: Recent, Home, Desktop, Documents, Downloads, Music, Pictures, Videos, Floppy Disk, and Other Locations. The main area shows a list of files with columns for Name, Size, and Modified. Two files are selected: 'Adware (004f7c2e1).zip' (779.8 kB, 18:52) and 'google-chrome-stable_current_amd64.deb' (96.7 MB, 03:24). At the top right are 'Reset' and 'Analyze' buttons. On the right, there's a 'Selection' panel with a search bar.

-Trên virtual box mở giao diện window lên , và file adware đang tác động lên window

Lab 1: Memory Forensic

9



-Dòng thông báo phân tích thành công

```
[cuckoo.core.scheduler] INFO: Task #1: reports generation completed
[cuckoo.core.scheduler] INFO: Task #1: analysis procedure completed
```

-Xem báo cáo từ mẫu phân tích

Task ID	Date	Filename / URL	Package	Status
1	23/08/2023 19:15	37ea273266aa2d28430194fc27849170d609d338ab9cfc45c4ebbe1bcf51f9 @ Adware {004f7c2e1}.zip	exe	Done ✓ reported

-Mẫu phân tích được đánh giá 3.6/10

Lab 1: Memory Forensic

10

The screenshot shows a 'Summary' section for an archive named 'Archive 37ea273266aa2d28430194fc2a27849170d609d338abc9c6c43c4e6be1bcf51f9'. The archive is a PE32 executable (GUI) Intel 80386, UPX compressed, with a size of 811.3KB. It has various hashes (MD5, SHA1, SHA256, SHA512, CRC32, ssdeep, Yara) and a Yara rule count of 0. A 'Score' box indicates numerous signs of malicious behavior with a score of 3.6 out of 10. A 'Feedback' section allows users to report different results.

-Một số phát hiện từ cuckoo

The screenshot shows a list of detected events from a Cuckoo analysis. The events include:

- Queries for the computername (1 event)
- Time & API (1 event)
- GetComputerNameW (1 event)
- Aug 23, 2023, 10:15 a.m.
- Resolves a suspicious Top Level Domain (TLD) (1 event)
- Foreign language identified in PE resource (10 events)
- Searches running processes potentially to identify processes for sandbox evasion, code injection or memory dumping (31 events)
- The binary likely contains encrypted or compressed data indicative of a packer (2 events)
- Potentially malicious URLs were found in the process memory dump (50 out of 127 events)
- The executable is compressed using UPX (2 events)
- Connects to an IP address that is no longer responding to requests (eg: remote services will remain up and running usually) (1 event)

-Chúng ta có thể export các báo cáo của cuckoo dưới dạng text:

The screenshot shows the 'Dashboard' page of the Cuckoo interface. On the left is a sidebar with various icons. The main area displays a list of files selected for export, including:

- extracted (0 files)
- files (0 files)
- memory (2 files)
- shots (0 files)
- reports (1 files)
- network (2 files)
- suricata (4 files)
- buffer (0 files)
- logs (2 files)
- dump.pcap
- analysis.log
- dump_sorted.pcap
- tismaster.txt
- task.json
- binary
- cuckoo.log
- files.json
- reboot.json

A button at the bottom says 'Download 11.7 MB'.

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

® Bài tập (yêu cầu làm)

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

Truy vấn python:

```

✓ [8] py_target_dir = "/content/drive/MyDrive/PythonSamples/"
      g = Github(username, password)
      py_repositories = g.search_repositories(query="language:python")
      n = 5
      i = 0

      for repo in py_repositories:
          reponame = repo.name
          target_dir_of_repo = py_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 == "py":
                          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

```

- Sau khi truy vấn ta có các mẫu python.

Lab 1: Memory Forensic

12

Name ↑	Owner	Last modified ▾	File size	⋮
∅ _init__.py	me	Apr 6, 2024 me	3 KB	⋮
∅ _init__.py	me	Apr 6, 2024 me	955 bytes	⋮
∅ _init__.py	me	Apr 6, 2024 me	0 bytes	⋮
∅ _main__.py	me	Apr 6, 2024 me	9 KB	⋮
∅ _main__.py	me	Apr 6, 2024 me	467 bytes	⋮
∅ _archive_maps.py	me	Apr 6, 2024 me	121 KB	⋮
∅ _config.py	me	Apr 6, 2024 me	532 bytes	⋮
∅ _test_bash_script.py	me	Apr 6, 2024 me	8 KB	⋮
∅ _test_finetune_rag.py	me	Apr 6, 2024 me	4 KB	⋮
∅ _test_make_student.py	me	Apr 6, 2024 me	2 KB	⋮

Truy vấn Powershell:

Lab 1: Memory Forensic

```
[9] ps_target_dir = "/content/drive/MyDrive/PowerShellSamples/"
    g = Github(username, password)
    ps_repositories = g.search_repositories(query="language:powershell")
    n = 5
    i = 0
```

```
▶ for repo in ps_repositories:
    reponame = repo.name
    target_dir_of_repo = ps_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 == "ps":
                    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
```

Tương tự với python, sau khi truy vấn powershell ta có các tập tin powershell.

Name	Owner	Last modified	File size	More
core	me	7:29 PM me	—	⋮
PowerSploit	me	7:31 PM me	—	⋮
Scoop	me	7:31 PM me	—	⋮
Windows10Debloater	me	7:31 PM me	—	⋮
WSL	me	7:31 PM me	—	⋮

f) Phân loại tập tin theo kiểu.

® Bài tập (yêu cầu làm)

5. Sinh viên cho biết quả của đoạn code trên

```

1m  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))

0.9825693996126533
[[ 456     7    11]
 [     6 1063     3]
 [     0     0     3]]

```

- Kết quả đoạn code trên cho thấy độ chính xác của mô hình khoảng 98.26%, nghĩa là khoảng 98,26% mẫu trong tập test đã được mô hình phân loại đúng.

- Như hình ta có thể thấy, 456 mẫu được phân loại javascript là đúng, 7 mẫu javascript được phân loại sai thành python, 11 mẫu được phân loại sai thành powershell. 1063 mẫu được phân loại python là đúng, 6 mẫu python được phân loại sai thành javascript, 3 mẫu python được phân loại sai thành powershell. 3 mẫu được phân loại powershell là đúng.

g) Đo lường sự giống nhau giữa hai chuỗi.

④ Bài tập (yêu cầu làm)

6. Sinh viên cho biết quả của đoạn code trên

- Cài đặt thư viện ssdeep.

```
cuong@cuong-ubuntu:~$ sudo apt-get install build-essential libffi-dev python3 python3-dev python3-pip automake autoconf libtool
[sudo] password for cuong:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.10.6-1~22.04).
python3 set to manually installed.
The following additional packages will be installed:
  autotools-dev binutils binutils-common binutils-x86-64-linux-gnu dpkg-dev
  fakeroot g++ g++-11 gcc gcc-11 javascript-common libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libbinutils
  libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0
  libctf0 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl
  libgcc-11-dev libitm1 libjs-jquery libjs-sphinxdoc libjs-underscore liblsan0
  libltdl-dev libnsl-dev libpython3-dev libpython3.10-dev libquadmath0
  libsigsegv2 libstdc++-11-dev libtirpc-dev libtsan0 libubsan1 linux-libc-dev
  lto-disabled-list m4 make manpages-dev python3-distutils python3-lib2to3
  python3-setuptools python3-wheel python3.10-dev rpcsvc-proto zlib1g-dev
Suggested packages:
  autoconf-archive gnu-standards autoconf-doc gettext binutils-doc
```

```
cuong@cuong-ubuntu:~$ sudo BUILD_LIB=1 pip install ssdeep
Collecting ssdeep
  Downloading ssdeep-3.4.tar.gz (110 kB)
    110.8/110.8 KB 1.9 MB/s eta 0:00:00
    Preparing metadata (setup.py) ... done
Collecting cffi>=0.8.6
  Using cached cffi-1.16.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (443 kB)
Requirement already satisfied: six>=1.4.1 in /usr/lib/python3/dist-packages (from ssdeep) (1.16.0)
Collecting pycparser
  Using cached pycparser-2.22-py3-none-any.whl (117 kB)
Building wheels for collected packages: ssdeep
  Building wheel for ssdeep (setup.py) ... done
    Created wheel for ssdeep: filename=ssdeep-3.4-cp310-cp310-linux_x86_64.whl size=53718 sha256=f9005c05b98ce97143a620d299c853f287329a2f735320895a9bf4d2b35793fe
    Stored in directory: /root/.cache/pip/wheels/29/9c/cb/04794e9a89fdec3acfb67930f12e99a727d1159f042b713c03
Successfully built ssdeep
Installing collected packages: pycparser, cffi, ssdeep
Successfully installed cffi-1.16.0 pycparser-2.22 ssdeep-3.4
```

Lab 1: Memory Forensic

```

1 import ssdeep
2
3 str1 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua."
4 str2 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua."
5 str3 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua."
6 str4 = "Something completely different from the other strings."
7
8 hash1 = ssdeep.hash(str1)
9 hash2 = ssdeep.hash(str2)
10 hash3 = ssdeep.hash(str3)
11 hash4 = ssdeep.hash(str4)
12 print(hash1)
13 print(hash2)
14 print(hash3)
15 print(hash4)
16
17 print(ssdeep.compare(hash1, hash1))
18 print(ssdeep.compare(hash1, hash2))
19 print(ssdeep.compare(hash1, hash3))
20 print(ssdeep.compare(hash1, hash4))

```

Bracket match found on line: 20 Python 2 ▾ Tab Width: 8 ▾ Ln 20, Col 36 ▾ INS

Kết quả sau khi chạy đoạn code trên, như ta có thể thấy mã hash str1 giống 39% mã hash str2, giống 37% so với mã hash str3, và khác hoàn toàn mã hash str4.

```

root@cuong-ubuntu:/home/cuong# python3 lab26.py
3:f4008MRwRJFGW1gC6uWv6MQ2MFSL+JuBF8BSnJi:f4kPvtHMCubyFtQ
3:f4008MRwRJFGW1gC6uWv6MQ2MFSL+JuBF8BS+EFECJi:f4kPvtHMCubyFIIsJQ
3:f4008MRwRJFGW1gC6uWv6MQ2MFSL+JuBF8BS6:f4kPvtHMCubyF0
3:60QKZ+4CDTfDaRFKYLVL:ywKDC2mVL
100
39
37
0

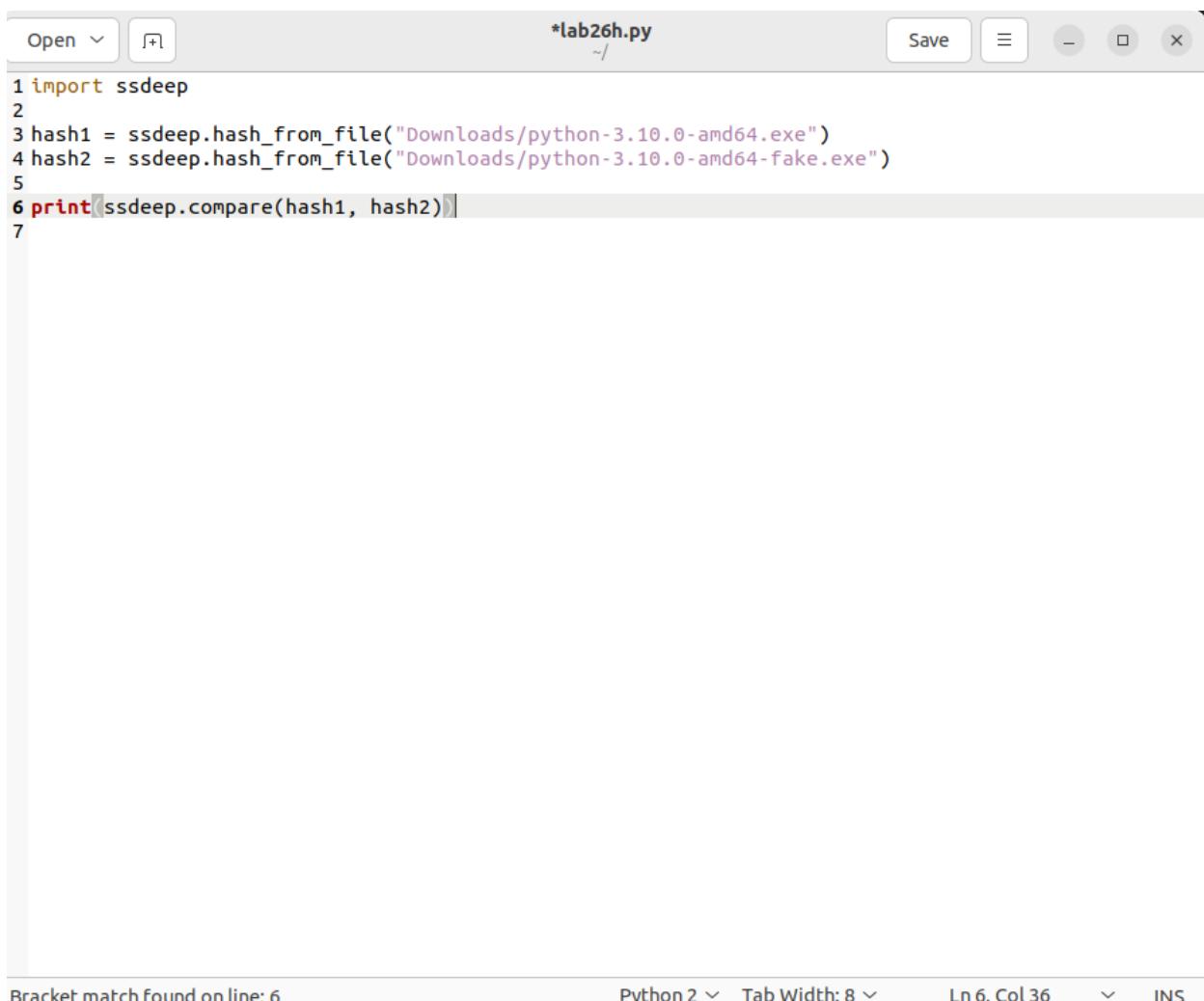
```

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

- Ta tạo 1 file copy python-3.10.0-amd64-fake.exe dựa trên file python-3.10.0-amd64.exe và thêm vài null bytes. Như hình dưới sau khi hexdump, file fake đã có sự khác biệt.

Lab 1: Memory Forensic

```
cuong@cuong-ubuntu:~/Downloads$ truncate -s +1 python-3.10.0-amd64-fake.exe
cuong@cuong-ubuntu:~/Downloads$ ls
burpsuite_community_linux_v2024_1_1_4.sh  python-3.10.0-amd64.exe
Burpsuite-Professional-2024-main          python-3.10.0-amd64-fake.exe
Burpsuite-Professional-2024-main.zip       text.ssd
oswap.tar                                 vuln-outdated-components-exercise.tar
cuong@cuong-ubuntu:~/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 00 00 00 00 00 00 00 00 00  |.....|
01b01118
cuong@cuong-ubuntu:~/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 00 00 00 00 00 00 00 00 00  |.....|
01b01119
```



```
*lab26h.py
~/

1 import ssdeep
2
3 hash1 = ssdeep.hash_from_file("Downloads/python-3.10.0-amd64.exe")
4 hash2 = ssdeep.hash_from_file("Downloads/python-3.10.0-amd64-fake.exe")
5
6 print(ssdeep.compare(hash1, hash2))
7
```

Bracket match found on line: 6 Python 2 Tab Width: 8 Ln 6, Col 36 INS

- Kết quả chạy đoạn code trên ta thấy không có sự khác biệt nào giữa 2 file python-3.10.0-amd64-fake.exe và python-3.10.0-amd64.exe.

Lab 1: Memory Forensic

```
cuong@cuong-ubuntu:~$ python3 lab26h.py
100
```

Điều đó cho thấy khi thêm null byte vào tệp python-3.10.0-amd64-fake.exe, các null byte không mang bất kỳ thông tin có ý nghĩa nào đối với nhiều loại tệp, vì vậy việc thêm chúng có thể không làm thay đổi đáng kể nội dung mà thuật toán ssdeep nhận biết. Vì vậy, ngay cả khi sửa đổi tệp python-3.10.0-amd64-fake.exe bằng cách thêm byte rỗng, độ tương tự về nội dung tổng thể vẫn có thể đủ cao để hàm ssdeep.compare trả về độ tương tự là 100.

i) Trích xuất N-Grams

® Bài tập (yêu cầu làm)
7. Sinh viên cho biết quả của đoạn code trên

```
✓ [15] pip install nltk
7s Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.12.25)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.2)

✓ [16] import collections
      from nltk import ngrams

✓ [17] file_to_analyze = "drive/MyDrive/Lab2/python-3.10.0-amd64.exe"

✓ [18] ⏎ #Ham doc byte cua tap tin
      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

✓ [19] ⏎ #Ham lay chuoi bytes va chuyen thanh ngrams
      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)
```

Lab 1: Memory Forensic

Lab 1: Memory Forensic

```

  ✓ 08 #Chon N-gram pho bien
    import numpy as np
    X = np.asarray(X)
    X_top_K2_freq = X[:, :K2]
    X_top_K2_freq

    [ 427928, 86437, 108028, 57985, 37707, 23, 27497,
      24224, 91987, 43216], [ 11322, 717, 1665, 1721, 1049, 0, 716,
      119, 1229, 784], [ 7969, 230, 543, 125, 436, 0, 251,
      56, 173, 138], [ 7670, 171, 590, 263, 449, 0, 354,
      84, 374, 225], [ 21528, 2305, 2779, 4933, 1921, 0, 1815,
      445, 837, 1083], [ 13045, 743, 2091, 2338, 1492, 0, 740,
      318, 464, 389], [ 4386, 42, 80, 67, 86, 0, 39,
      10, 116, 47], [ 30694, 4419, 2415, 4993, 2102, 9, 1124,
      1894, 1807, 674], [ 8054, 117, 594, 336, 376, 0, 153,
      76, 987, 249], [ 3628, 175, 0, 94, 2, 0, 0,
      69, 84, 0], [ 17578, 2488, 3, 421, 10, 15, 3,
      598, 539, 1], [ 5507, 471, 1, 121, 12, 0, 1,
      69, 97, 0], [ 58780, 2185, 0, 220, 18, 73, 1],
  
```



```

  ✓ 38 #Dua vao 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],
       [ 38, 2, 2, 4, 2, 13, 2, 17, 1,  0],
       [ 81, 4, 1, 0, 3, 3, 7, 8, 1, 368],
       [ 0, 3, 0, 11, 2, 28, 2, 7, 1,  0],
       [ 1, 5, 0, 17, 5, 17, 2, 11, 0,  0],
       [ 162, 8, 3, 5, 4, 23, 15, 51, 2, 736],
       [ 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, 18, 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, 28, 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],
       [ 3, 19, 11, 6, 105, 25, 23, 30, 40,  0],
  
```



```

  ✓ #Chi-squared algorithm
    chi2_selector = SelectKBest(chi2, k=K2)
    X_top_K2_chi2 = chi2_selector.fit_transform(X, y)
    X_top_K2_chi2

    array([[ 615, 433, 574, 335, 46, 552, 84, 4, 7,
           1],
       [ 171, 32, 125, 6, 11, 46, 1, 4, 0,
           9],
       [ 19, 0, 10, 1, 3, 6, 0, 0, 0,
           6],
       [ 436, 447, 80, 8, 623, 12, 1, 6, 0,
           0],
       [ 72, 127, 126, 48, 30, 81, 27, 5, 14,
           1],
       [ 16, 22, 42, 9, 5, 13, 5, 1, 7,
           1],
       [ 149, 222, 147, 90, 54, 121, 40, 3, 24,
           0],
       [ 62, 191, 218, 104, 57, 61, 11, 4, 11,
           0],
       [ 75, 192, 165, 72, 57, 54, 19, 3, 27,
           2],
       [ 9, 26, 26, 7, 3, 14, 1, 0, 1,
           1],
       [ 130, 115, 78, 6, 71, 71, 1, 0, 1,
           0],
       [ 131, 10, 19, 1, 8, 2, 2, 3, 0,
           0],
       [ 90, 6, 12, 1, 7, 2, 1, 1, 0,
           0],
       [ 294, 69, 74, 54, 37, 6, 7, 10, 1,
           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.

[®] Bài tập (yêu cầu làm)

8. Sinh viên hoàn thành các bước trên

Bước 1: Tạo mẫu và gán nhãn

```
[73] import os
from os import listdir

[74] directories_and_labels = [("drive/MyDrive/Lab2/Benign PE Samples", 0),
("drive/MyDrive/Lab2/Malicious PE Samples", 1)]
list_of_samples = []
labels = []
N_spec = 2

[75] 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)

[76] 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
['drive/MyDrive/Lab2/Benign PE Samples/junction.exe', 'drive/MyDrive/Lab2/Benign PE Samples/ls.exe', 'drive/MyDrive/Lab2/Benign PE Samples/last.exe', 'drive/MyDrive/Lab2/Benign PE Samples/malicious.exe', 'drive/MyDrive/Lab2/Benign PE Samples/7ZipSetup.exe', 'drive/MyDrive/Lab2/Malicious PE Samples/win32.exe', 'drive/MyDrive/Lab2/Malicious PE Samples/Fake Intel (R) Dual Band Wireless-AC 7265.exe', 'drive/MyDrive/Lab2/Malicious PE Samples/Windows 10 Pro (x64).exe', 'drive/MyDrive/Lab2/Malicious PE Samples/Windows 10 Pro (x64).exe']

[77] 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]
[1, 1, 1, 1, 1]
```

Bước 2: Chia dữ liệu thành tập train và test

Lab 1: Memory Forensic

```

[64] from sklearn.model_selection import train_test_split

[65] samples_train, samples_test, target_train, target_test = train_test_split(
...     list_of_samples,
...     labels,
...     test_size=0.3,
...     stratify=labels,
...     random_state=11
... )

[ ]

```

Bước 3: Xây dựng các hàm lấy thuộc tính

```

[79] import collections
      from nltk import ngrams
      import numpy as np
      import pefile

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

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

[82] 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)

[83] 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

```

```

[84] 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)

[85] 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)

[86] 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)

```

Bước 4: Chọn 100 thuộc tính phô biến với 2-grams

Lab 1: Memory Forensic

```

1m  Ngrams_count_all = collections.Counter([])
2s    for sample in samples_train:
3s      Ngrams_count_all += bin_file_to_Ngrams_count(sample, N_spec)
4s    K1 = 100
5s    K1_most_common_Ngrams = Ngrams_count_all.most_common(K1)
6s    K1_most_common_Ngrams_list = [x[0] for x in K1_most_common_Ngrams]

```

Bước 5:

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

```

1s  [88] imports_corpus_train = []
2s    num_sect_train = []
3s    sect_name_train = []
4s    Ngram_feat_list_train = []

5s    y_train = []

6s
7m  [89] for i in range(len(samples_train)):
8s    sample = samples_train[i]
9s    try:
10s      # Get all required parameters with predefined functions
11s      Ngram_features = get_Ngrams_features_from_samples(sample, K1_most_common_Ngrams_list)
12s      pe = pefile.PE(sample)
13s      imports = get_imports(pe)
14s      n_sections = len(pe.sections)
15s      sec_names = get_section_names(pe)

16s      # Put above value into lists
17s      imports_corpus_train.append(imports)
18s      num_sect_train.append(n_sections)
19s      sect_name_train.append(sec_names)
20s      Ngram_feat_list_train.append(Ngram_features)

21s      # Target train
22s      y_train.append(target_train[i])
23s    except Exception as e:
24s      print(sample + ":")
25s      print(e)

```

Bước 6: Sử dụng hàm băm tfidf để chuyển imports, section names từ văn bản thành dạng số

```

1s  [90] from sklearn.feature_extraction.text import HashingVectorizer, TfidfTransformer
2s    from sklearn.pipeline import Pipeline

3s
4s  [91] imports_featurizer = Pipeline(
5s    [
6s      ("vect", HashingVectorizer(input = "content", ngram_range=(1,2))),
7s      ("tfidf", TfidfTransformer(use_idf = True)),
8s    ]
9s  )

10s [92] sect_name_featurizer = Pipeline(
11s    [
12s      ("vect", HashingVectorizer(input = "content", ngram_range= (1,2))),
13s      ("tfidf", TfidfTransformer(use_idf = True))
14s    ]
15s  )

16s [93] imports_corpus_train_transformed = imports_featurizer.fit_transform(imports_corpus_train)
17s
18s [94] sect_name_train_transformed = sect_name_featurizer.fit_transform(sect_name_train)

```

Lab 1: Memory Forensic

Bước 7: Kết hợp các vector thuộc tính thành 1 mảng.

```
✓ 0s [95] from scipy.sparse import hstack, csr_matrix

✓ 0s [96] X_train = hstack(
    [
        Ngram_feat_list_train,
        imports_corpus_train_transformed,
        sect_name_train_transformed,
        csr_matrix(num_sect_train).transpose(),
    ]
)
```

Bước 8: Ta huấn luyện bằng phân loại Random Forest cho tập train

```
✓ 0s [98] from sklearn.ensemble import RandomForestClassifier

✓ 9s [99] clf = RandomForestClassifier(n_estimators = 100)
         clf = clf.fit(X_train, y_train)
```

Bước 9: Thu thập các thuộc tính của tập test, giống như tập huấn luyện

```
✓ 0s [100] import_corpus_test = []
           num_sect_test = []
           sect_names_test = []
           Ngram_feat_list_test = []

           y_test = []

1m 0s ⏎ 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)

☒ drive/MyDrive/Lab2/Benign PE Samples/CasPol.exe:
'DOS Header magic not found.'
drive/MyDrive/Lab2/Benign PE Samples/CasPol.exe:
'DOS Header magic not found.'
drive/MyDrive/Lab2/Benign PE Samples/CasPol.exe:
'DOS Header magic not found.'
drive/MyDrive/Lab2/Benign PE Samples/CasPol.exe:
'PE' object has no attribute 'DIRECTORY_ENTRY_IMPORT'
drive/MyDrive/Lab2/Benign PE Samples/CasPol.exe:
'DOS Header magic not found.'
drive/MyDrive/Lab2/Benign PE Samples/CasPol.exe:
```

```
0s    y_test
0s      0,
      1,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      1,
      0,
      0,
      0,
```

Bước 10: Ta chuyển đổi vector từ thuộc tính test, và kiểm tra kết quả của trình phân loại.

```
[104] import_corpus_test_transformed = imports_featurizer.transform(import_corpus_test)

[105] sect_names_test_transformed = imports_featurizer.transform(sect_names_test)

[106] X_test = hstack(
    [
        Ngram_feat_list_test,
        import_corpus_test_transformed,
        sect_names_test_transformed,
        csr_matrix(num_sect_test).transpose()
    ]
)

[107] print("The score of our classifier is as follow: ")
      print(clf.score(X_test, y_test))

The score of our classifier is as follow:
0.91
```

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File **.DOCX** và **.PDF**. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng **Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)** – cỡ **chữ 13**. **Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.**
- Đặt tên theo định dạng: [Mã lớp]-ExeX_GroupY. (trong đó X là Thứ tự Bài tập, Y là mã số thứ tự nhóm trong danh sách mà GV phụ trách công bố).

Ví dụ: [NT101.K11.ANTT]-Exe01_Group03.

- Nếu báo cáo có nhiều file, nén tất cả file vào file **.ZIP** với cùng tên file báo cáo.
- **Không đặt tên đúng định dạng – yêu cầu, sẽ KHÔNG chấm điểm bài nộp.**
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.

HẾT