

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN**



Báo cáo bài thực hành

Phân loại mã độ

sinh viên thực hiện:

B20DCAT129 Chu Minh Nghĩa

Giảng viên hướng dẫn: TS. Nguyễn Ngọc Điệp

MỤC LỤC

MỤC LỤC	1
DANH MỤC CÁC HÌNH VẼ.....	2
DANH MỤC CÁC BẢNG BIỂU	2
DANH MỤC CÁC TỪ VIẾT TẮT.....	3
1.1 Giới thiệu chung về bài thực hành	4
1.2 Nội dung và hướng dẫn bài thực hành	4
1.2.1 Mục đích.....	4
1.2.2 Yêu cầu đối với sinh viên.....	4
1.2.3 Nội dung thực hành	4
1.3 Phân tích yêu cầu bài thực hành	9
1.4 Thiết kế bài thực hành	10
1.5 Cài đặt và cấu hình các máy ảo	12
1.6 Tích hợp và triển khai	14
1.6.1 Docker Hub	14
1.6.2 Github.....	15
1.7 Thử nghiệm và đánh giá.....	17
TÀI LIỆU THAM KHẢO.....	20

DANH MỤC CÁC HÌNH VẼ

Hình 1 Giao diện Labedit của bài lab.....	12
Hình 2 Cấu hình file tretaslocal.....	12
Hình 3 Cài đặt phần Result	13
Hình 4 Cài đặt phần Goals	13
Hình 5 Dockerfiles của máy malbasic.....	13
Hình 6 Add và commit bài lab	14
Hình 7 Đẩy các vùng chứa lên dockerhub	14
Hình 8 Các vùng chứa được đẩy lên dockerhub	15
Hình 9 Tạo file Imodule.tar.....	15
Hình 10 File imodule.tar chứa bài thực hành.....	15
Hình 11 Đẩy file imodule.tar lên github	16
Hình 12 Lấy được lab được lưu trữ về.....	16
Hình 13 Các tệp cần thiết trong máy malbasic.....	17
Hình 14 Chạy lệnh python3 hash_file.py	17
Hình 15 Kết quả phân tích VirusTotal	18
Hình 16 Ghi kết quả vào results.txt.....	18
Hình 17 In kết quả results.txt	19
Hình 18 Chạy lệnh classify.py	19
Hình 19 Đánh giá kết quả bài thực hành.....	19

DANH MỤC CÁC BẢNG BIỂU

Bảng 1. Bảng Result.....	10
Bảng 2. Bảng Goals.....	11

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Thuật ngữ tiếng Anh/Giải thích	Thuật ngữ tiếng Việt/Giải thích
Bsic	Basic	Cơ sở
Mal	Malware	Mã độc
API	Application programming interface	Giao diện lập trình ứng dụng, cung cấp cách thức cho các chương trình giao tiếp với nhau.
API KEY	Application Programming Interface key	Một chuỗi ký tự đặc biệt được nhà cung cấp API cung cấp cho người dùng để xác thực và quản lý quyền truy cập vào API của họ.

1.1 Giới thiệu chung về bài thực hành

Bài thực hành "Phân loại mã độc" được thiết kế nhằm giúp sinh viên hiểu rõ hơn về các loại mã độc phổ biến, đặc điểm nhận diện và cách thức hoạt động của chúng. Đây là một bước khởi đầu quan trọng để sinh viên làm quen với các kỹ thuật liên quan đến bảo mật thông tin và hiểu sâu hơn về các mối đe dọa an ninh mạng hiện nay. Bài thực hành này hướng dẫn sinh viên sử dụng VirusTotal API để kiểm tra mã hash và phân loại mã độc như virus, trojan, worm, ransomware, và spyware [1]. Sinh viên sẽ làm quen với việc tích hợp API vào Python script để gửi yêu cầu và nhận kết quả phân tích. Qua đó, sinh viên sẽ học cách trích xuất thông tin quan trọng như loại mã độc và các chỉ số nhận diện khác. Việc thực hành này không chỉ giúp sinh viên nắm vững cách sử dụng công cụ, mà còn phát triển kỹ năng tự động hóa các tác vụ phân tích bảo mật.

1.2 Nội dung và hướng dẫn bài thực hành

1.2.1 Mục đích

Giúp sinh viên tìm hiểu về các loại mã độc phổ biến, cách phân loại và nhận diện chúng thông qua việc sử dụng VirusTotal API.

1.2.2 Yêu cầu đối với sinh viên

Sinh viên cần có kiến thức cơ bản về hệ điều hành Linux và lập trình Python, bao gồm cách làm việc với giao diện dòng lệnh, viết script, và sử dụng các thư viện hỗ trợ gửi yêu cầu HTTP. Đồng thời, sinh viên cũng nên hiểu cơ bản về cách sử dụng API để tích hợp vào các ứng dụng phân tích bảo mật.

1.2.3 Nội dung thực hành

Khởi động bài lab:

Vào terminal, gõ:

```
Labtainer -r mal_bsic_define
```

(chú ý: sinh viên sử dụng mã sinh viên của mình để nhập thông tin email người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm)

Sau khi khởi động xong một terminal ảo sẽ xuất hiện.

Bài lab đã có sẵn thư mục `malware_samples` chứa 15 mẫu mã độc. Chúng ta sẽ thực hiện chỉnh sửa script trong `hash_file.py` để lấy mã hash của các mẫu trong thư mục `malware_samples` rồi chạy lệnh, sinh viên có thể dùng code mẫu ở dưới:

```
python3 hash_file.py
```

```

import hashlib
import os

def calculate_hash(file_path, hash_type='md5'):
    """Tính hash của một file."""
    hash_func = hashlib.new(hash_type)
    try:
        with open(file_path, 'rb') as f:
            while True: # Đọc từng khối dữ liệu
                chunk = f.read(8192)
                if not chunk: # Nếu không còn dữ liệu, thoát vòng lặp
                    break
                hash_func.update(chunk)
            return hash_func.hexdigest()
    except FileNotFoundError:
        print(f"Không tìm thấy file: {file_path}")
        return None

def hash_files_in_directory(directory, hash_type='md5'):
    """Tính hash cho tất cả các file trong thư mục."""
    file_hashes = {}
    for root, _, files in os.walk(directory):
        for file in files:
            file_path = os.path.join(root, file)
            file_hash = calculate_hash(file_path, hash_type)
            if file_hash:
                # Lưu tên file không có đuôi và hash
                file_name_without_extension = os.path.splitext(file)[0]

```

```

        file_hashes[file_name_without_extension] = file_hash
    return file_hashes

def extract_number(name):
    """Trích xuất số từ tên file để sắp xếp."""
    return int(''.join(filter(str.isdigit, name)) or 0)

# Thư mục chứa mẫu mã độc
directory = "./malware_samples"

# Tính hash cho các file
file_hashes = hash_files_in_directory(directory)

# Sắp xếp dựa trên số trong tên file và in tên file cùng hash
for name, hash in sorted(file_hashes.items(), key=lambda x: extract_number(x[0])):
    print(f"{name}: {hash}")

```

Câu lệnh ghi kết quả ra terminal hoành thành task1. Sinh viên copy mã hash của từng mẫu vào trang <https://www.virustotal.com/gui/home/search> để xem phân tích, Tiếp theo sinh viên ghi kết quả phân tích của từng mẫu vào results.txt theo cấu trúc dưới đây ghi đủ 15 mẫu với lệnh:

nano results.txt

```

sample1: <mã hash 1> <loại mã độc đặc trưng nhất trên virustotal>
sample2: <mã hash 2> <loại mã độc đặc trưng nhất trên virustotal>
sample3: <mã hash 3> <loại mã độc đặc trưng nhất trên virustotal>

```

Dùng lệnh cat để in ra kết quả vừa ghi:

cat results.txt

Sau khi in đầy đủ các mẫu mã độc và phân loại tương ứng của chúng coi như đã hoàn thành task2.

Tiếp theo nhiệm vụ của sinh viên là đăng ký tài khoản VirusTotal và lấy API KEY, thực hiện chỉnh sửa script trong classify.py thêm API KEY vừa tạo rồi chạy lệnh, sinh viên có thể dùng code mẫu ở dưới:

python3 classify.py

```
import requests
import hashlib
import os
import re

# Thay đổi YOUR_API_KEY bằng giá trị của bạn
API_KEY = 'YOUR_API_KEY'

def calculate_hash(file_path, hash_type='md5'):
    """Tính MD5 hash của một file."""
    hash_func = hashlib.new(hash_type)
    try:
        with open(file_path, 'rb') as f:
            while True:
                chunk = f.read(8192)
                if not chunk:
                    break
                hash_func.update(chunk)
        return hash_func.hexdigest()
    except FileNotFoundError:
        print(f"Không tìm thấy file: {file_path}")
        return None
```



```

def hash_files_in_directory(directory, hash_type='md5'):
    """Tính hash cho tất cả các file trong thư mục."""
    file_hashes = {}
    for root, _, files in os.walk(directory):
        for file in files:
            file_path = os.path.join(root, file)
            file_hash = calculate_hash(file_path, hash_type)
            if file_hash:
                # Lấy tên file không có phần mở rộng
                file_name = os.path.splitext(file)[0]
                file_hashes[file_name] = file_hash
    return file_hashes

def get_threat_category(file_hash):
    """Gửi yêu cầu đến VirusTotal API để lấy thông tin phân loại mã độc."""
    url = f'https://www.virustotal.com/api/v3/files/{file_hash}'
    headers = {'x-apikey': API_KEY}
    response = requests.get(url, headers=headers)

    if response.status_code == 200:
        analysis_result = response.json()
        popular_threats = analysis_result.get("data", {}).get("attributes",
        {}).get("popular_threat_classification", {}).get("popular_threat_category",
        [])
        if popular_threats:
            most_common_threat = max(popular_threats, key=lambda x: x['count'])
            return most_common_threat['value']
    return None

def natural_sort_key(s):
    """Tạo khóa sắp xếp tự nhiên từ chuỗi."""
    return [int(text) if text.isdigit() else text.lower() for text in
    re.split('(\d+)', s)]

```

```

# Thư mục chứa mẫu mã độc (đặt cùng cấp với script Python)
directory = "./malware_samples"

# Tính hash cho các file
file_hashes = hash_files_in_directory(directory)

# Sắp xếp file_hashes theo tên file với sắp xếp tự nhiên
sorted_hashes = sorted(file_hashes.items(), key=lambda x:
    natural_sort_key(x[0]))

# Gửi yêu cầu API và in kết quả từng dòng
for file_name, file_hash in sorted_hashes:
    threat_category = get_threat_category(file_hash)
    if threat_category:
        print(f"{file_name}: {file_hash} {threat_category}")

```

Câu lệnh ghi kết quả là các mẫu mã độc và phân loại tương ứng ra terminal hoàn thành task3.

Kết thúc bài lab:

Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

stoplab mal_bsic_define

1.3 Phân tích yêu cầu bài thực hành

Bài thực hành được thực hiện trên một container với môi trường được cấu hình sẵn, bao gồm các tệp cần thiết như `classify.py`, `hash_file.py`, thư mục `malware_samples` chứa các mẫu mã độc, và tệp kết quả `results.txt`. Sinh viên sẽ thực hiện các nhiệm vụ như chỉnh sửa script để lấy mã hash của các tệp trong thư mục mẫu, phân tích mã độc thủ công bằng cách sử dụng VirusTotal, và tích hợp API KEY vào script để tự động phân loại mã độc. Qua bài thực hành này, sinh viên sẽ làm quen với quy trình phân tích mã độc trong một môi trường an toàn và được kiểm soát, đồng

thời phát triển kỹ năng lập trình Python và sử dụng API để tự động hóa các tác vụ phân tích bảo mật.

1.4 Thiết kế bài thực hành

Trên môi trường máy ảo Ubuntu được cung cấp, sử dụng docker tạo ra 1 container: container mang tên “malbasic”

Cấu hình docker gồm có:

- malbasic: lưu cấu hình cho máy:
 - Tên máy: malbasic
- config: lưu cấu hình hoạt động của hệ thống
- dockerfiles: mô tả cấu hình của 1 container trong đó:
 - malbasic: sử dụng các thư viện mặc định hệ thống.

Các nhiệm vụ cần phải thực hiện để thực hành thành công:

- Thực hiện chỉnh sửa script trong hash_file.py để lấy mã hash của các mẫu trong thư mục malware_samples.
- Thực hiện copy mã hash của từng mẫu vào virustotal để xem phân tích, Tiếp theo sinh viên ghi kết quả phân tích của từng mẫu vào results.txt.
- Nhiệm vụ của sinh viên là đăng ký tài khoản VirusTotal và lấy API KEY, thực hiện chỉnh sửa script trong classify.py thêm API KEY vừa tạo để lấy phân loại tự động các mẫu trong thư mục malware_samples.

Kết thúc bài lab và đóng gói kết quả.

Để đánh giá được sinh viên đã hoàn thành bài thực hành hay chưa, cần chia bài thực hành thành các nhiệm vụ nhỏ, mỗi nhiệm vụ cần phải chỉ rõ kết quả để có thể dựa vào đó đánh giá, chấm điểm. Do vậy, trong bài thực hành này hệ thống cần ghi nhận các thao tác, sự kiện được mô tả và cấu hình như bảng 1,2:

Bảng 1. Bảng Result

Result Tag	Container	File	Field Type	Field ID	Timestamp Type
------------	-----------	------	------------	----------	----------------

_hash1	malbasic	hash_file.py.stdout	CONTAINS	sample1: 9f8bc96c96d43 ecb69f883388d 228754	File
_class1	malbasic	cat.stdout	CONTAINS	sample1: 9f8bc96c96d43 ecb69f883388d 228754 ransomware	File
_pclas1	malbasic	classify.py.stdout	CONTAINS	sample1: 9f8bc96c96d43 ecb69f883388d 228754 ransomware	File

- _hash1: hiển thị mã hash của mẫu mã độc 1 ra terminal qua việc chạy hash_file.py, tương tự với _hash2, _hash3 đến _hash15.

- _class1: hiển thị mã hash của mẫu mã độc 1 ra terminal cùng với phân loại tương ứng của nó qua lệnh cat, tương tự với _class2, _class3 đến _class15.

- _pclas1: hiển thị mã hash của mẫu mã độc 1 ra terminal cùng với phân loại tương ứng của nó qua việc chạy classify.py, tương tự với _pclas2, _pclas3 đến _pclas15.

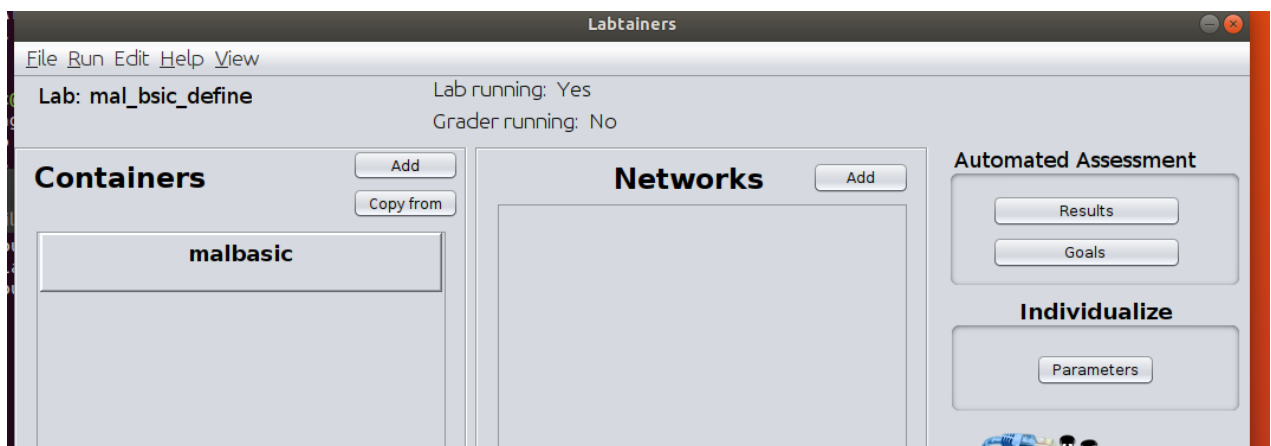
Bảng 2. Bảng Goals

Goal ID	Goal Type	Boolean
hash	boolean	_hash1 and _hash2 and _hash3 and _hash4 and _hash5 and _hash6 and _hash7 and _hash8 and _hash9 and _hash10 and _hash11 and _hash12 and _hash13 and _hash14 and _hash15
classify	boolean	_class1 and _class2 and _class3 and _class4 and _class5 and _class6 and _class7 and _class8 and _class9 and _class10 and _class11 and _class12 and _class13 and _class14 and _class15

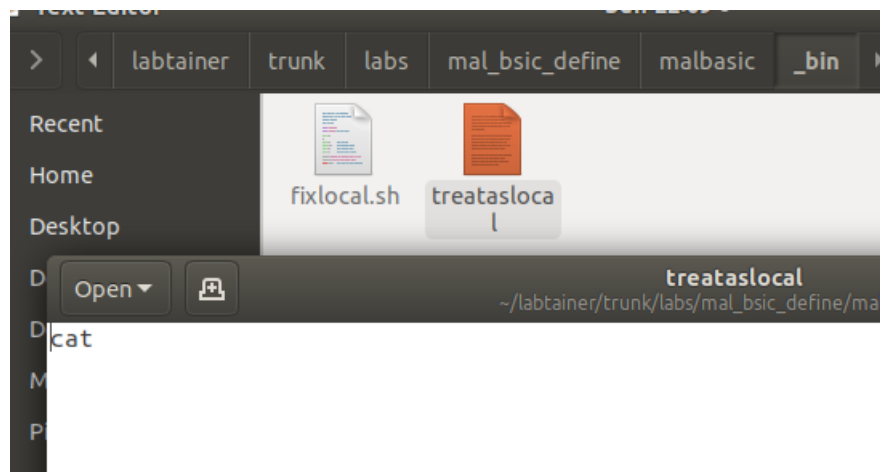
py_classify	boolean	_pclas1 and _pclas2 and _pclas3 and _pclas4 and _pclas5 and _pclas6 and _pclas7 and _pclas8 and _pclas9 and _pclas10 and _pclas11 and _pclas12 and _pclas13 and _pclas14 and _pclas15
-------------	---------	--

- hash: đúng khi tất cả _hash1, _hash2,..., hash15 đều đúng.
- classify: đúng khi tất cả _class1, _class2,..., _class15 đều đúng.
- py_classify: đúng khi tất cả _pclas1, _pclas2,..., _pclas15 đều đúng.

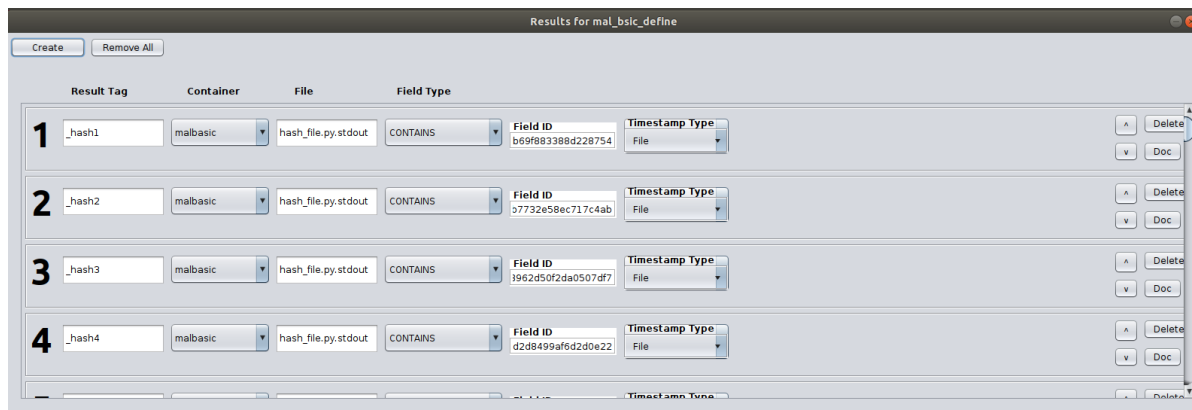
1.5 Cài đặt và cấu hình các máy ảo



Hình 1 Giao diện Labedit của bài lab



Hình 2 Cấu hình file tretaslocal



Hình 3 Cài đặt phân Result



Hình 4 Cài đặt phân Goals



Hình 5 Dockerfiles của máy malbasic

1.6 Tích hợp và triển khai

Bài thực hành đã được triển khai như sau:

1.6.1 Docker Hub

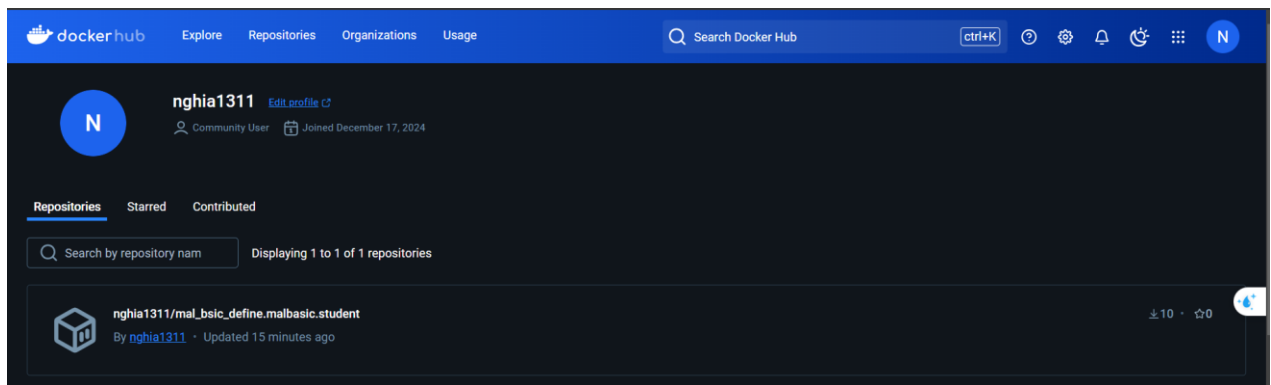
<https://hub.docker.com/r/nghia1311>

```
student@ubuntu:~/labtainer/trunk/labs$ git add mal_bsic_define
student@ubuntu:~/labtainer/trunk/labs$ git commit mal_bsic_define -m "Adding an Imodule"
error: pathspec 'an' did not match any file(s) known to git.
error: pathspec 'Imodule'" did not match any file(s) known to git.
student@ubuntu:~/labtainer/trunk/labs$ git commit mal_bsic_define -m "Adding an Imodule"
[master (root-commit) 1860e73] Adding an Imodule
14 files changed, 564 insertions(+)
create mode 100644 mal_bsic_define/config/malbasic-home_tar.list
create mode 100644 mal_bsic_define/config/parameter.config
create mode 100644 mal_bsic_define/config/start.config
create mode 100644 mal_bsic_define/dockerfiles/Dockerfile.mal_bsic_define.malbasic.student
create mode 100644 mal_bsic_define/docs/read_first.txt
create mode 100644 mal_bsic_define/instr_config/goals.config
create mode 100755 mal_bsic_define/instr_config/pregrade.sh
create mode 100644 mal_bsic_define/instr_config/results.config
create mode 100755 mal_bsic_define/malbasic/_bin/fixlocal.sh
create mode 100644 mal_bsic_define/malbasic/_bin/treataslocal
create mode 100644 mal_bsic_define/malbasic/_system/etc/login.defs
create mode 100644 mal_bsic_define/malbasic/_system/etc/securetty
create mode 100644 mal_bsic_define/malbasic/home_tar/home.tar
create mode 100644 mal_bsic_define/malbasic/sys_tar/sys.tar
student@ubuntu:~/labtainer/trunk/labs$
```

Hình 6 Add và commit bài lab

```
Login Succeeded
student@ubuntu:~/labtainer/trunk/distrib$ ./publish.py -d -l mal_bsic_define
adding [nmaplab]
adding [httplab]
adding [liveforensics]
adding [bind-shell]
adding [tlab]
adding [metasploitable-test]
adding [kali-test]
adding [my-remote-dns]
adding [remote-dns2]
adding [remote-dns]
adding [backups]
adding [centos-log]
adding [dhcp-test]
adding [xlab]
adding [softplc]
adding [iptables]
adding [grfics]
adding [usbttest]
adding [ida]
adding [centossix]
adding [routing-basics2]
adding [shellbasics]
```

Hình 7 Đẩy các vùng chứa lên dockerhub



Hình 8 Các vùng chứa được đẩy lên dockerhub

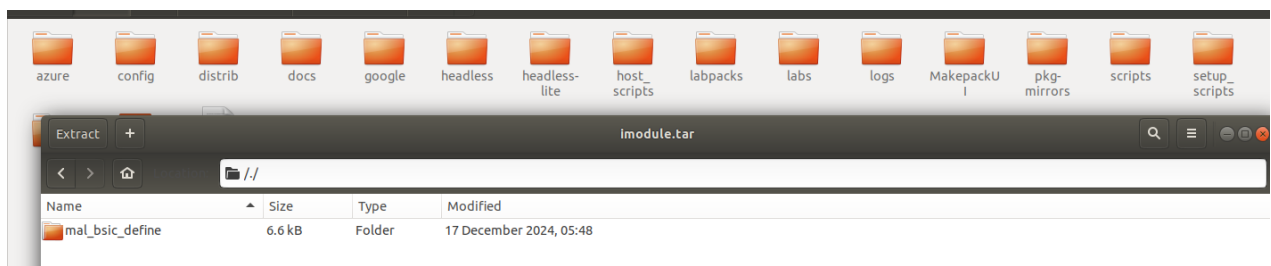
1.6.2 Github

https://github.com/cnghia1311/mal_lab

Nhập lệnh create-imodules.sh

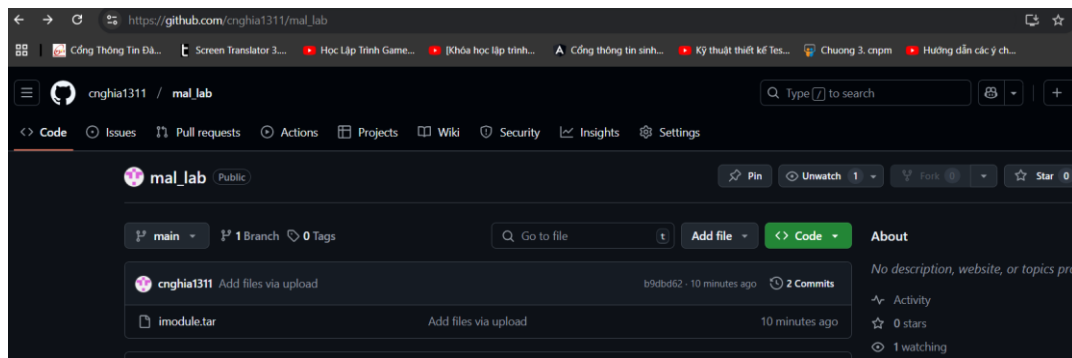
```
student@ubuntu:~/labtainer/trunk/labs$ create-imodules.sh
lab is mal_bsic_define
Do docs
*****
** Post /home/student/labtainer/trunk/imodule.tar to your web server **
*****
student@ubuntu:~/labtainer/trunk/labs$
```

Hình 9 Tạo file Imodule.tar

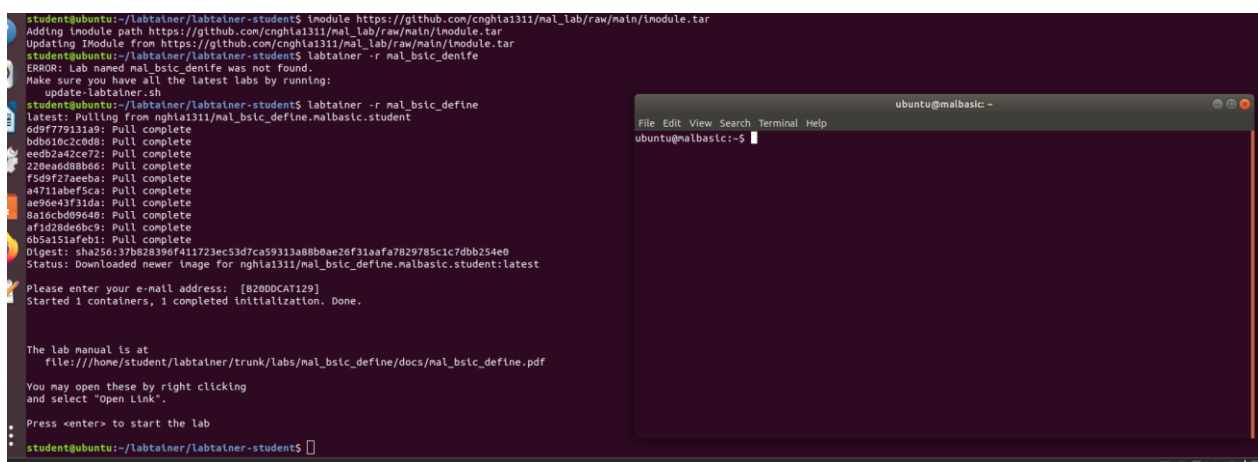


Hình 10 File imodule.tar chứa bài thực hành

Tạo repo mới để đẩy imodule.tar lên và tạo phần release mới



Hình 11 Đẩy file imodule.tar lên github

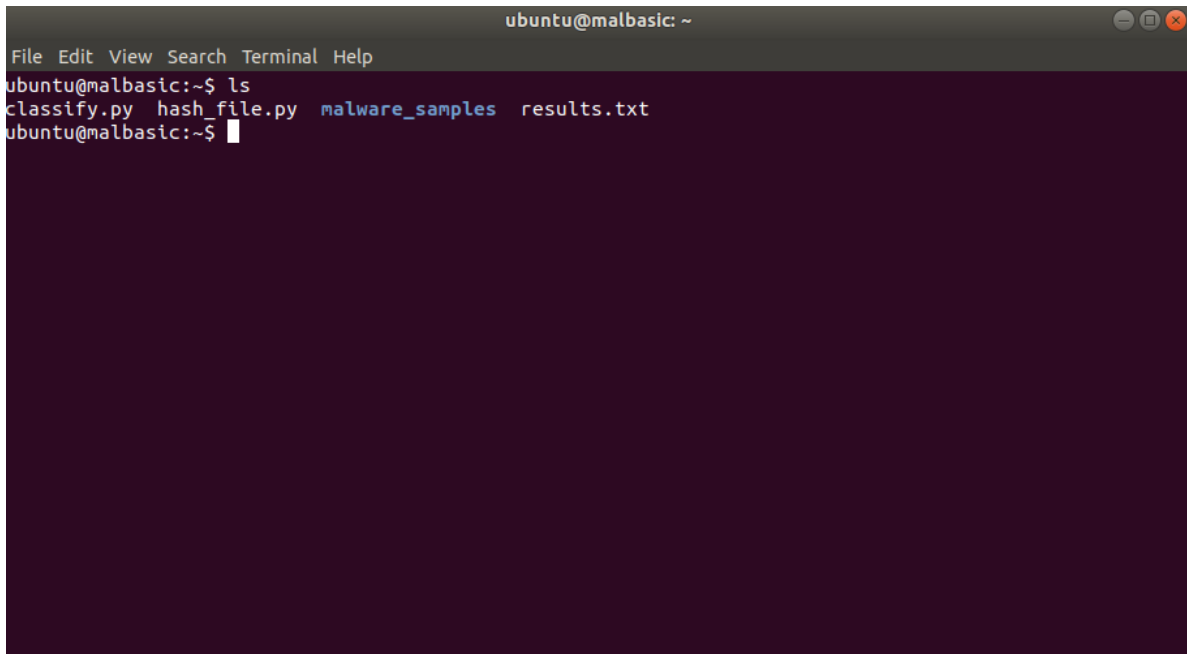


Hình 12 Lấy được lab được lưu trữ về

1.7 Thử nghiệm và đánh giá

Bài thực hành đã được xây dựng thành công, dưới đây là hình ảnh minh họa về bài thực hành:

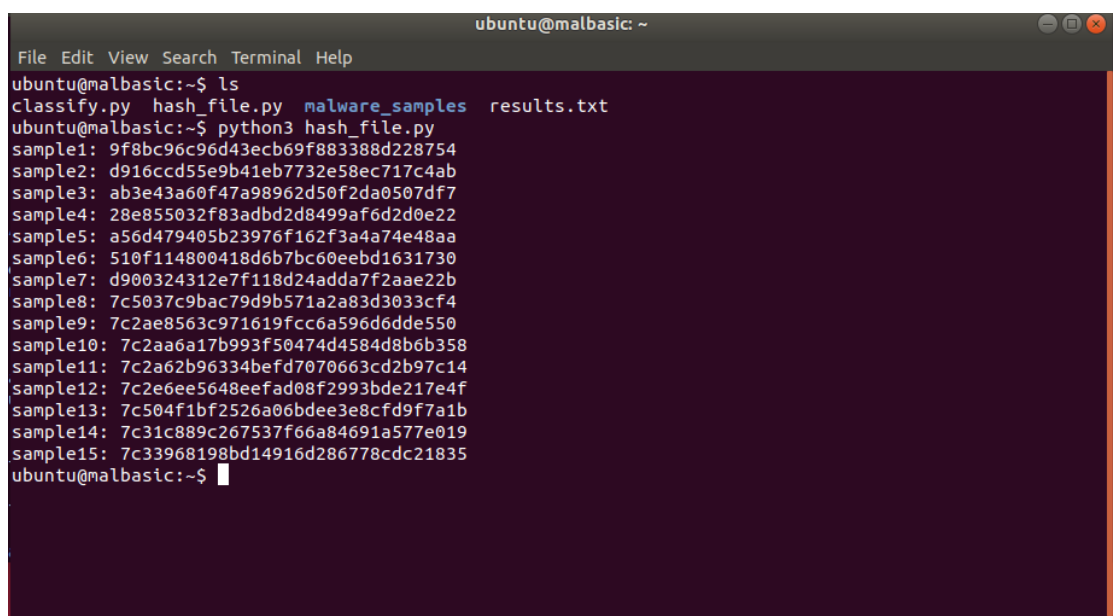
- Bài lab đã có sẵn thư mục `malware_samples` chứa 15 mẫu mã độc. Sinh viên dùng lệnh `ls` để kiểm tra.



```
ubuntu@malbasic: ~  
File Edit View Search Terminal Help  
ubuntu@malbasic:~$ ls  
classify.py hash_file.py malware_samples results.txt  
ubuntu@malbasic:~$
```

Hình 13 Các tệp cần thiết trong máy malbasic

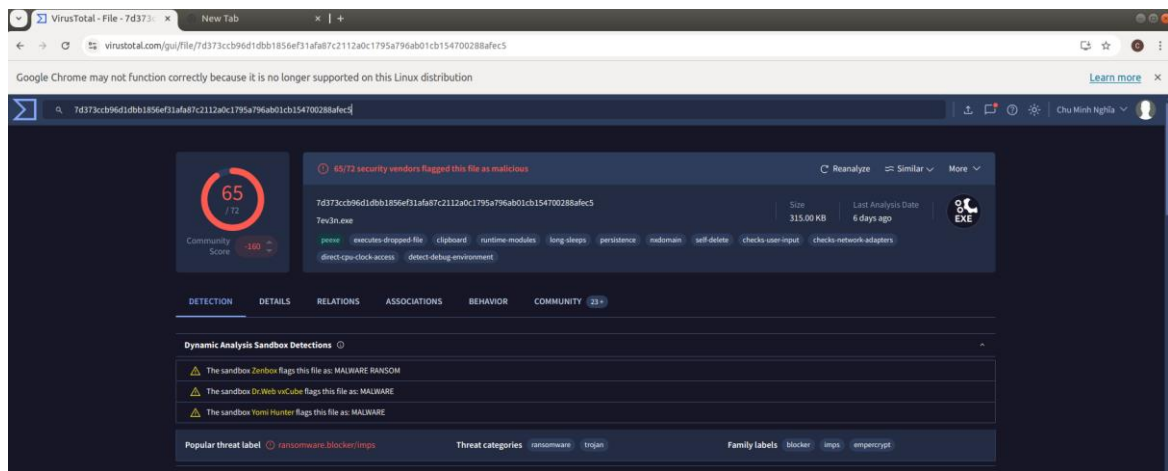
- Thực hiện chỉnh sửa script trong `hash_file.py` để lấy mã hash của các mẫu trong thư mục `malware_samples` rồi chạy lệnh `python3 hash_file.py`.



```
ubuntu@malbasic: ~  
File Edit View Search Terminal Help  
ubuntu@malbasic:~$ ls  
classify.py hash_file.py malware_samples results.txt  
ubuntu@malbasic:~$ python3 hash_file.py  
sample1: 9f8bc96c96d43ecb69f883388d228754  
sample2: d916ccd55e9b41eb7732e58ec717c4ab  
sample3: ab3e43a60f47a98962d50f2da0507df7  
sample4: 28e855032f83adbd2d8499af6d2d0e22  
sample5: a56d479405b23976f162f3a4a74e48aa  
sample6: 510f114800418d6b7bc60eebd1631730  
sample7: d900324312e7f118d24adda7f2aae22b  
sample8: 7c5037c9bac79d9b571a2a83d3033cf4  
sample9: 7c2ae8563c971619fcc6a596d6dde550  
sample10: 7c2aa6a17b993f50474d4584d8b6b358  
sample11: 7c2a62b96334befd7070663cd2b97c14  
sample12: 7c2e6ee5648eefad08f2993bde217e4f  
sample13: 7c504f1bf2526a06bdee3e8cfd9f7a1b  
sample14: 7c31c889c267537f66a84691a577e019  
sample15: 7c33968198bd14916d286778cdc21835  
ubuntu@malbasic:~$
```

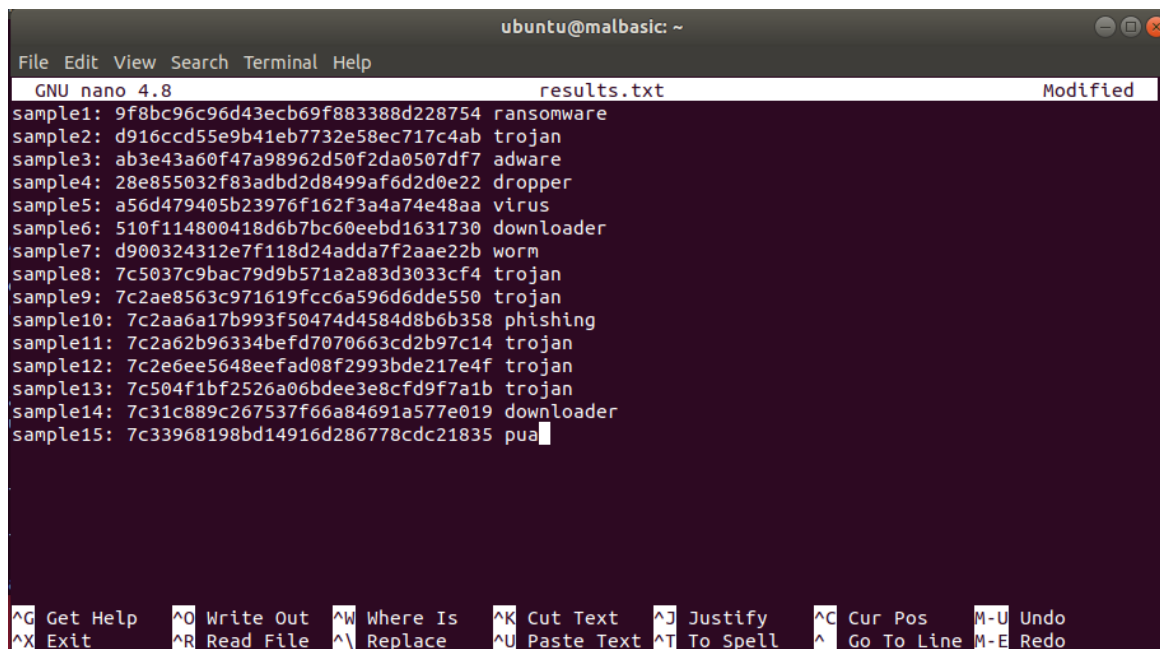
Hình 14 Chạy lệnh `python3 hash_file.py`

- Sinh viên copy mã hash của từng mẫu vào trang virustotal để xem phân tích.



Hình 15 Kết quả phân tích VirusTotal

- Tiếp theo sinh viên ghi kết quả phân tích của từng mẫu vào results.txt ghi đủ 15 mẫu.



Hình 16 Ghi kết quả vào results.txt

- Dùng lệnh cat để in ra kết quả vừa ghi.

```
ubuntu@malbasic:~$ cat results.txt
sample1: 9f8bc96c96d43ecb69f883388d228754 ransomware
sample2: d916ccd55e9b41eb7732e58ec717c4ab trojan
sample3: ab3e43a60f47a98962d50f2da0507df7 adware
sample4: 28e855032f83adbd2d8499af6d2d0e22 dropper
sample5: a56d479405b23976f162f3a4a74e48aa virus
sample6: 510f114800418d6b7bc60eebd1631730 downloader
sample7: d900324312e7f118d24adda7f2aae22b worm
sample8: 7c5037c9bac79d9b571a2a83d3033cf4 trojan
sample9: 7c2ae8563c971619fcc6a596d6dde550 trojan
sample10: 7c2aa6a17b993f50474d4584d8b6b358 phishing
sample11: 7c2a62b96334befd7070663cd2b97c14 trojan
sample12: 7c2e6ee5648eefad08f2993bde217e4f trojan
sample13: 7c504f1bf2526a06bdee3e8cf9f7a1b trojan
sample14: 7c31c889c267537f66a84691a577e019 downloader
sample15: 7c33968198bd14916d286778cdc21835 pua
ubuntu@malbasic:~$
```

Hình 17 In kết quả results.txt

- Tiếp theo nhiệm vụ của sinh viên là đăng ký tài khoản VirusTotal và lấy API KEY, thực hiện chỉnh sửa script trong classify.py thêm API KEY vừa tạo rồi chạy lệnh python3 classify.py Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab

```
ubuntu@malbasic:~$ python3 classify.py
sample1: 9f8bc96c96d43ecb69f883388d228754 ransomware
sample2: d916ccd55e9b41eb7732e58ec717c4ab trojan
sample3: ab3e43a60f47a98962d50f2da0507df7 adware
sample4: 28e855032f83adbd2d8499af6d2d0e22 dropper
sample5: a56d479405b23976f162f3a4a74e48aa virus
sample6: 510f114800418d6b7bc60eebd1631730 downloader
sample7: d900324312e7f118d24adda7f2aae22b worm
sample8: 7c5037c9bac79d9b571a2a83d3033cf4 trojan
sample9: 7c2ae8563c971619fcc6a596d6dde550 trojan
sample10: 7c2aa6a17b993f50474d4584d8b6b358 phishing
sample11: 7c2a62b96334befd7070663cd2b97c14 trojan
sample12: 7c2e6ee5648eefad08f2993bde217e4f trojan
sample13: 7c504f1bf2526a06bdee3e8cf9f7a1b trojan
sample14: 7c31c889c267537f66a84691a577e019 downloader
sample15: 7c33968198bd14916d286778cdc21835 pua
ubuntu@malbasic:~$
```

Hình 18 Chạy lệnh classify.py

- Trên terminal đầu tiên sử dụng câu lệnh *checkwork* để xem kết quả bài lab

```
student@ubuntu:~/labtainer/labtainer-student$ checkwork
Results stored in directory: /home/student/labtainer_xfer/mal_bsic_define
Labname mal_bsic_define

Student | hash | classify | py_classify |
=====|=====|=====|=====|
B20DDCAT129 | Y | Y | Y |
What is automatically assessed for this lab:
```

Hình 19 Đánh giá kết quả bài thực hành

TÀI LIỆU THAM KHẢO

- [1] M. K. A. Monnappa, Learning Malware Analysis, Birmingham, UK: Packt Publishing, 2018, pp. 30-34.