

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



Báo cáo bài thực hành

Hướng dẫn sử dụng công cụ UPX, Ghidra và Yara

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	6
1.4 Thiết kế bài thực hành	6
1.5 Cài đặt và cấu hình các máy ảo	8
1.6 Tích hợp và triển khai	9
1.6.1 Docker Hub	10
1.6.2 Github.....	11
1.7 Thử nghiệm và đánh giá.....	12
TÀI LIỆU THAM KHẢO.....	15

DANH MỤC CÁC HÌNH VẼ

Hình 1 Giao diện Labedit của bài lab.....	8
Hình 2 Cài đặt phần Result	9
Hình 3 Dockerfiles của máy maltool.....	9
Hình 4 Add và commit bài lab	10
Hình 5 Đẩy các vùng chứa lên dockerhub	10
Hình 6 Các vùng chứa được đẩy lên dockerhub	11
Hình 7 Tạo file Imodule.tar.....	11
Hình 8 File imodule.tar chứa bài thực hành	11
Hình 9 Đẩy file imodule.tar lên github	12
Hình 10 Lấy lab lưu trữ về	12
Hình 11 Các tệp cần thiết trong máy maltol.....	12
Hình 12 Chạy lệnh upx -t sample1.exe	13
Hình 13 Chạy lệnh upx -d sample1.exe	13
Hình 14 Tạo thành công project Ghidra và tìm được đầy đủ thông điệp.....	13
Hình 15 Tạo file rule	14
Hình 16 Viết rule.....	14
Hình 17 Chạy lệnh yara sample1_rule.yar sample1.exe	14
Hình 18 Đánh giá kết quả bài thực hành	14

DANH MỤC CÁC BẢNG BIỂU

Bảng 1. Bảng Result.....	7
--------------------------	---

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Thuật ngữ tiếng Anh/Giải thích	Thuật ngữ tiếng Việt/Giải thích
Mal	Malware	Mã độc
UPX	Ultimate Packer for eXecutables	Công cụ nén tệp thực thi để giảm kích thước, thường được sử dụng bởi mã độc để làm khó quá trình phân tích.
	Repository	Một kho lưu trữ được sử dụng để quản lý và tổ chức các tệp, thường liên quan đến một dự án phần mềm. Một repository chứa toàn bộ các tệp của dự án, bao gồm mã nguồn, tài liệu, và lịch sử thay đổi của dự án theo thời gian.
	Rule	Quy tắc được sử dụng để nhận diện và phân loại.

1.1 Giới thiệu chung về bài thực hành

Bài thực hành "mal_tool" được thiết kế nhằm giúp sinh viên làm quen với các công cụ phổ biến trong việc phân tích mã độc, qua đó hiểu rõ hơn về cách thức hoạt động và phương pháp nhận diện các loại mã độc. Đây là một bước quan trọng để sinh viên nắm bắt các kỹ thuật phân tích mã độc và cải thiện kỹ năng bảo mật thông tin.

Bài lab này giúp sinh viên làm quen với ba công cụ chính: **UPX**, **Ghidra**, và **Yara**, hoạt động trên hệ điều hành Linux. UPX được sử dụng để giải nén các tệp thực thi đã bị nén [2], tạo điều kiện thuận lợi cho việc phân tích chi tiết hơn với Ghidra [1]. Ghidra là một công cụ dịch ngược mã máy, cung cấp phân tích tĩnh, cho phép sinh viên tìm hiểu cấu trúc và hành vi của mã độc. Trong khi đó, Yara được sử dụng để xác định và phân loại mã độc dựa trên các mẫu hoặc quy tắc (rules) do người dùng định nghĩa.

Thông qua bài thực hành, sinh viên sẽ học cách sử dụng các công cụ này trong quy trình phân tích mã độc một cách hiệu quả. Bên cạnh đó, bài lab còn giúp sinh viên phát triển kỹ năng thực tế, như giải nén tệp mã độc, thực hiện phân tích tĩnh và xây dựng quy tắc nhận diện mã độc dựa trên các đặc điểm đã phân tích.

Bài thực hành này không chỉ giúp sinh viên hiểu rõ hơn về việc sử dụng công cụ, mà còn cung cấp nền tảng để đối phó với các mối đe dọa an ninh mạng trong các tình huống thực tế.

1.2 Nội dung và hướng dẫn bài thực hành

1.2.1 Mục đích

Giúp sinh viên tìm hiểu về các công cụ phân tích mã độc phổ biến, bao gồm UPX, Ghidra, và Yara, để giải nén, phân tích tĩnh, và nhận diện mã độc. Qua đó, sinh viên sẽ hiểu cách sử dụng các công cụ này trong thực tế để phân loại và nhận diện các loại mã độc dựa trên các mẫu và quy tắc do người dùng định nghĩa.

1.2.2 Yêu cầu đối với sinh viên

Sinh viên cần có kiến thức cơ bản về hệ điều hành Linux, bao gồm kỹ năng làm việc với giao diện dòng lệnh và thao tác với các công cụ phân tích mã độc như UPX, Ghidra, và Yara. Đồng thời, sinh viên nên hiểu cách tạo và sử dụng các quy tắc Yara để nhận diện mã độc và cách thực hiện phân tích tĩnh mã độc bằng Ghidra.

1.2.3 Nội dung thực hành

Khởi động bài lab:

Vào terminal, gõ:

Labtainer -r mal_tool

(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, máy ảo đã có sẵn thư mục Ghidra và mẫu mã độc sample1.

Task 1: Kiểm tra file thực thi có được nén bằng UPX, sinh viên chạy lệnh:

upx -t sample1.exe

Nếu tệp được nén bằng UPX, terminal sẽ hiển thị thông báo xác nhận.

Task 2: Giải nén tệp thực thi bằng UPX, sinh viên chạy lệnh:

upx -d sample1.exe

Sau khi giải nén, tệp sẽ sẵn sàng để thực hiện các phân tích chi tiết hơn.

Task 3: Phân tích mã nhị phân bằng Ghidra, chuyển đến thư mục Ghidra và khởi chạy:

cd ghidra

./ghidraRun

Tiếp theo thực hiện các bước:

- Chọn **File** → **New Project**.
- Chọn **Non-Shared Project** và đặt tên cho dự án (ví dụ: sample1).
- Sau khi dự án được tạo, nhấn **OK**.
- Tiếp theo, chọn **File** → **Import File**.
- Chọn tệp sample1.exe và nhấn **OK** để import tệp vào dự án.
- Sau khi file được nhập, nhấn **Analyze** để bắt đầu phân tích.
- Khi giao diện CodeBrowser hiện lên, vào Window → Define Strings.
- Lọc từ khóa "**encrypt**" trong ô filter để tìm các chuỗi có liên quan.
- Ghi thông điệp đầy đủ dòng "YOUR PERSONAL ..." hiển thị vào tệp:

nano results.txt

Task 4: Tạo rules và phân loại mã độc bằng Yara, tạo tệp rule Yara dựa trên thông điệp đầy đủ đã ghi nhận. Ví dụ:

touch sample1_rule.yar

nano sample1_rule.yar

```
rule Sample1_Rule {  
  strings:  
    $msg = "YOUR PERSONAL ..."  
  condition:  
    $msg  
}
```

Lưu tệp rule (ví dụ: sample1_rule.yar) và chạy lệnh:

yara sample1_rule.yar sample1.exe

Nếu thông điệp được phát hiện trong sample1.exe, thông báo thành công sẽ hiển thị.

Kết thúc bài lab:

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

stoplab mal_tool

1.3 Phân tích yêu cầu bài thực hành

Bài thực hành này được thực hiện trên một máy tính ảo duy nhất, với môi trường đã được cấu hình sẵn, bao gồm các công cụ phân tích mã độc như UPX, Ghidra, Yara, và các tệp cần thiết như các mẫu mã độc trong thư mục malware_samples cùng các tệp kết quả như results.txt. Sinh viên sẽ thực hiện các nhiệm vụ như kiểm tra xem các tệp có bị nén bằng UPX hay không, sử dụng UPX để giải nén các tệp mã độc, phân tích mã nhị phân bằng Ghidra để tìm hiểu cấu trúc và hành vi của mã độc, và tạo các quy tắc Yara để phân loại mã độc dựa trên kết quả phân tích. Qua bài thực hành này, sinh viên sẽ làm quen với quy trình phân tích mã độc sử dụng các công cụ phổ biến trong bảo mật, đồng thời phát triển kỹ năng làm việc với các công cụ phân tích bảo mật, phân tích tĩnh mã độc và tạo quy tắc Yara để nhận diện mã độc.

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 “maltool”

Cấu hình docker gồm có:

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

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

- Kiểm tra xem mẫu mã độc sample1.exe có bị nén bằng UPX hay không bằng cách sử dụng lệnh `upx -t sample1.exe`.
- Sử dụng UPX để giải nén mẫu mã độc bằng lệnh `upx -d sample1.exe`.
- Tiến hành phân tích mã nhị phân của mẫu mã độc đã giải nén bằng công cụ Ghidra. Sinh viên cần tìm kiếm các từ khóa liên quan, đặc biệt là chuỗi có chứa thông điệp "YOUR PERSONAL ...". Sau đó, ghi lại thông điệp đầy đủ này vào tệp results.txt
- Dựa trên kết quả đã phân tích từ Ghidra, tạo các quy tắc Yara để nhận diện và phân loại mẫu mã độc. Sau khi viết file quy tắc, sử dụng công cụ Yara để kiểm tra lại mẫu mã độc sample1.exe và xác nhận kết quả nhận diện.

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:

Bảng 1. Bảng Result

Result Tag	Container	File	Field Type	Field ID	Timestamp Type
checkupx	maltool	upx.stdout	CONTAINS	Tested 1 file.	File

unpack	maltool	upx.stdout	CONTAINS	Unpacked 1 file.	File
ghidra	maltool	results.txt	CONTAINS	YOUR PERSONAL INFORMATION ARE ENCRYPTED by 7ev3n	File
yara	maltool	yara.stdout	CONTAINS	Sample1_Rule sample1.exe	File

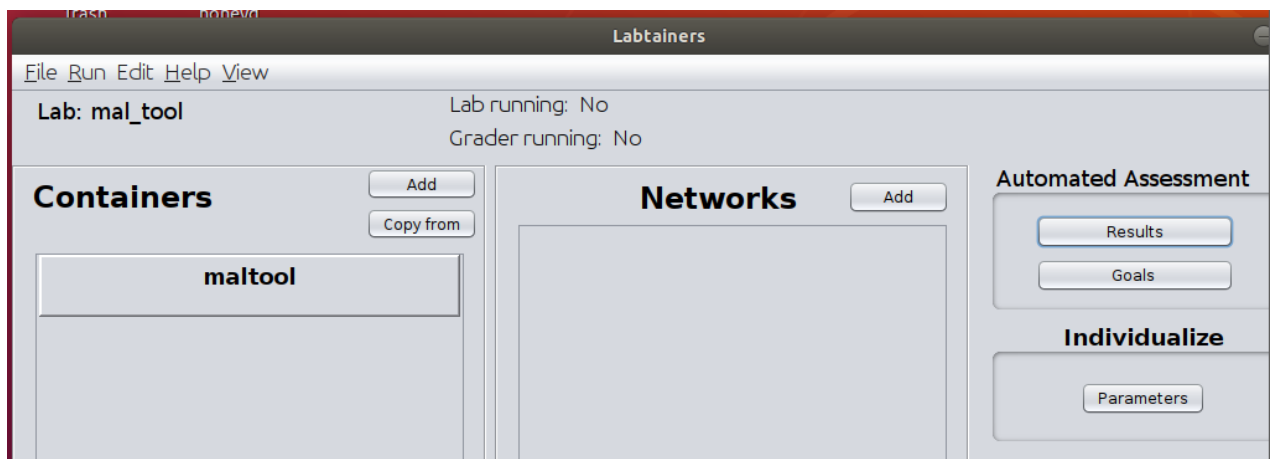
- `checkupx`: hiển thị thông báo mẫu mã độc đã được nén bằng UPX trước đó trên terminal qua việc chạy lệnh kiểm tra `upx -t sample1.exe`.

- `unpack`: hiển thị thông báo mẫu mã độc đã được giải nén thành công bằng UPX qua lệnh `upx -d sample1.exe`.

- `ghidra`: thông điệp được ghi đầy đủ vào `results.txt`.

- `yara`: hiển thị đã phát hiện thông điệp trong `sample1.exe` sau khi kiểm tra bằng lệnh `yara sample1_rule.yar sample1.exe`.

1.5 Cài đặt và cấu hình các máy ảo



Hình 1 Giao diện Labedit của bài lab

	Result Tag	Container	File	Field Type	Field ID	Timestamp Type
1	checkupx	maltool	upx.stdout	CONTAINS	Tested 1 file.	File
2	unpack	maltool	upx.stdout	CONTAINS	Unpacked 1 file.	File
3	ghidra	maltool	results.txt	CONTAINS	ENCRYPTED by 7ev3n	File
4	yara	maltool	yara.stdout	CONTAINS	e1_Rule sample1.exe	File

Hình 2 Cài đặt phần Result

```

ARG registry
FROM $registry/labtainer.base2
#FROM $registry/labtainer.network
#FROM $registry/labtainer.centos
#FROM $registry/labtainer.lamp
#
# lab is the fully qualified image name, e.g., mylab.some_container.student
# labdir is the name of the lab, e.g., mylab
# imagedir is the name of the container
# user_name is the USER from the start.config, if other than ubuntu,
# then that user must be added in this dockerfile
# before the USER command
#
ARG lab
ARG labdir
ARG imagedir
ARG user_name
ARG password
ARG apt_source
ARG version
LABEL version=$version
ENV APT_SOURCE $apt_source
RUN /usr/bin/apt-source.sh

# Cài đặt JDK 17
RUN apt-get update
RUN apt-get install -y openjdk-17-jdk
RUN apt-get install -y upx
RUN apt-get install -y yara
#
# put package installation here, e.g.,
# RUN apt-get update && apt-get install -y --no-install-recommends somepackage
  
```

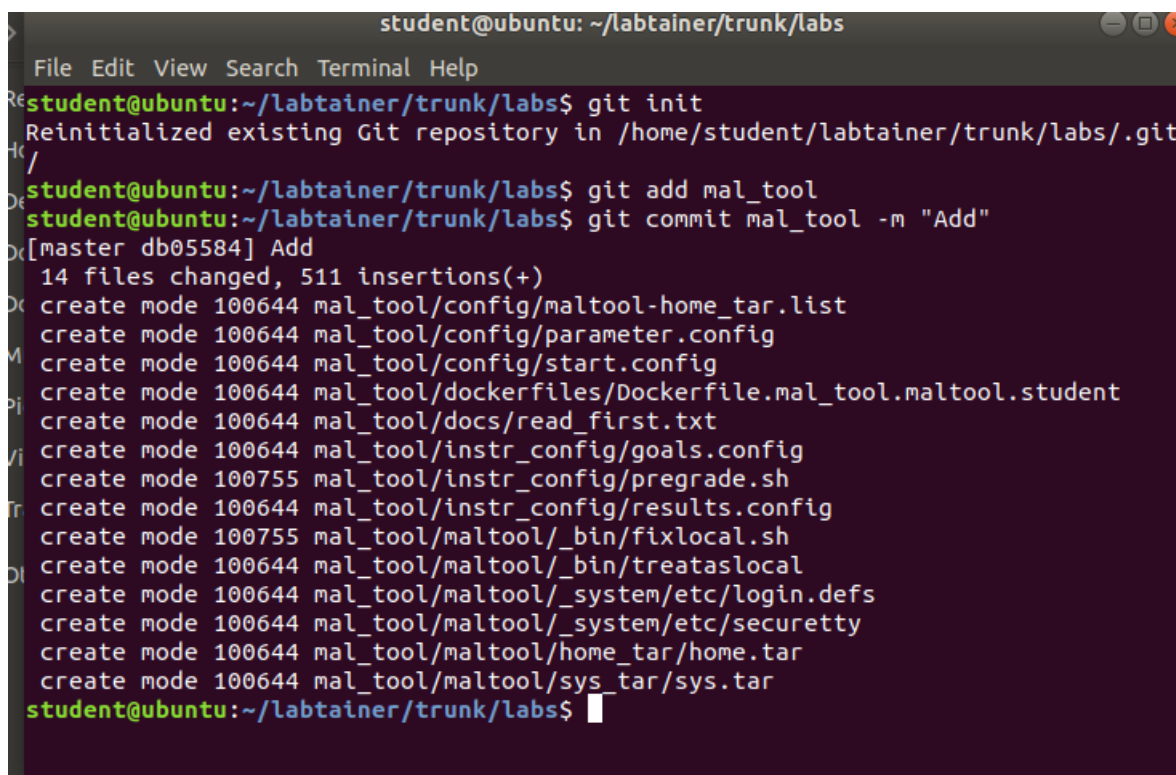
Hình 3 Dockerfiles của máy maltool

1.6 Tích hợp và triển khai

Bài thực hành đã được triển khai như sau:

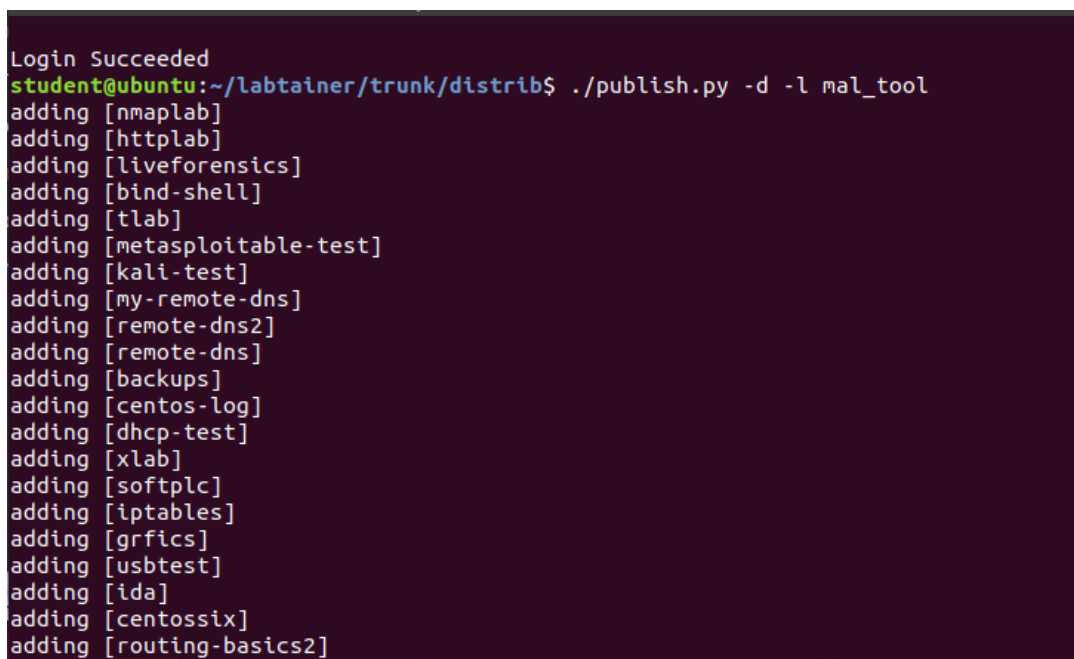
1.6.1 Docker Hub

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



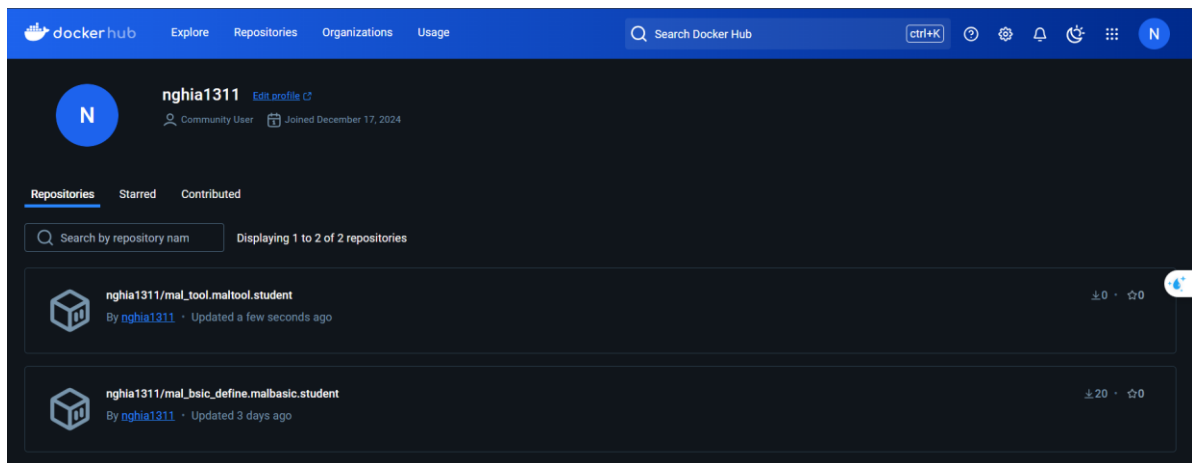
```
student@ubuntu: ~/labtainer/trunk/labs
File Edit View Search Terminal Help
student@ubuntu:~/labtainer/trunk/labs$ git init
Reinitialized existing Git repository in /home/student/labtainer/trunk/labs/.git
student@ubuntu:~/labtainer/trunk/labs$ git add mal_tool
student@ubuntu:~/labtainer/trunk/labs$ git commit mal_tool -m "Add"
[master db05584] Add
14 files changed, 511 insertions(+)
create mode 100644 mal_tool/config/maltool-home_tar.list
create mode 100644 mal_tool/config/parameter.config
create mode 100644 mal_tool/config/start.config
create mode 100644 mal_tool/dockerfiles/Dockerfile.mal_tool.maltool.student
create mode 100644 mal_tool/docs/read_first.txt
create mode 100644 mal_tool/instr_config/goals.config
create mode 100755 mal_tool/instr_config/pregrade.sh
create mode 100644 mal_tool/instr_config/results.config
create mode 100755 mal_tool/maltool/_bin/fixlocal.sh
create mode 100644 mal_tool/maltool/_bin/treataslocal
create mode 100644 mal_tool/maltool/_system/etc/login.defs
create mode 100644 mal_tool/maltool/_system/etc/securetty
create mode 100644 mal_tool/maltool/home_tar/home.tar
create mode 100644 mal_tool/maltool/sys_tar/sys.tar
student@ubuntu:~/labtainer/trunk/labs$
```

Hình 4 Add và commit bài lab



```
Login Succeeded
student@ubuntu:~/labtainer/trunk/distrib$ ./publish.py -d -l mal_tool
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 [usbtest]
adding [ida]
adding [centossix]
adding [routing-basics2]
```

Hình 5 Đẩy các vùng chứa lên dockerhub



Hình 6 Các vùng chứa được đẩy lên dockerhub

1.6.2 Github

https://github.com/cnghia1311/mal_lab

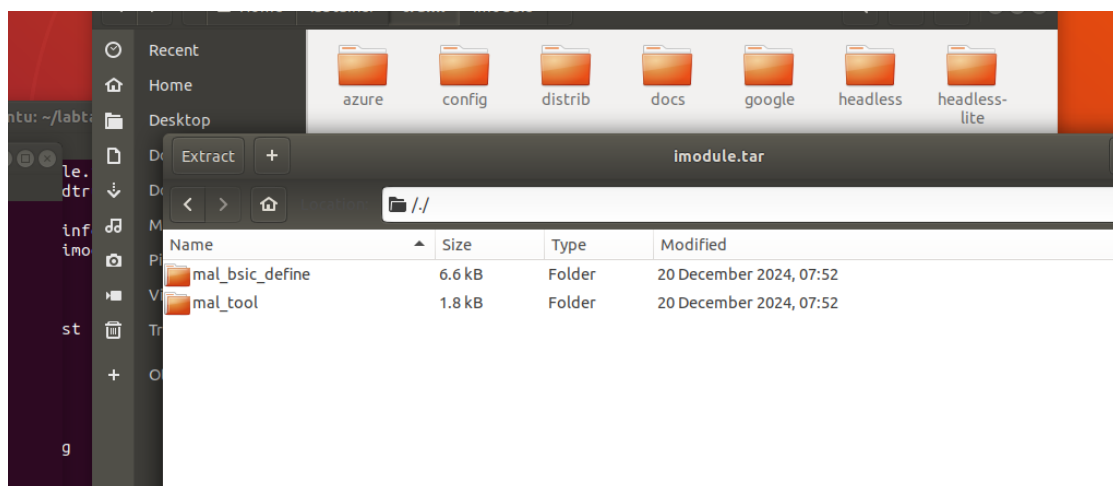
Nhập lệnh create-imodules.sh

```

Create mode 100644 mal_tool/maltool/sys_mal_tool.maltool.student.tar.gz
student@ubuntu:~/labtainer/trunk/labs$ create-imodules.sh
lab is mal_bsic_define
Do docs
lab is mal_tool
Do docs
*****
** Post /home/student/labtainer/trunk/imodule.tar to your web server **
*****
student@ubuntu:~/labtainer/trunk/labs$

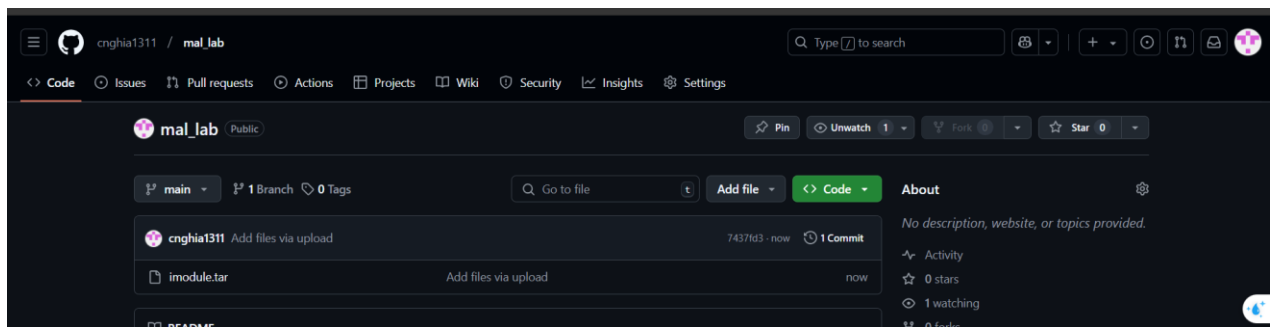
```

Hình 7 Tạo file Imodule.tar

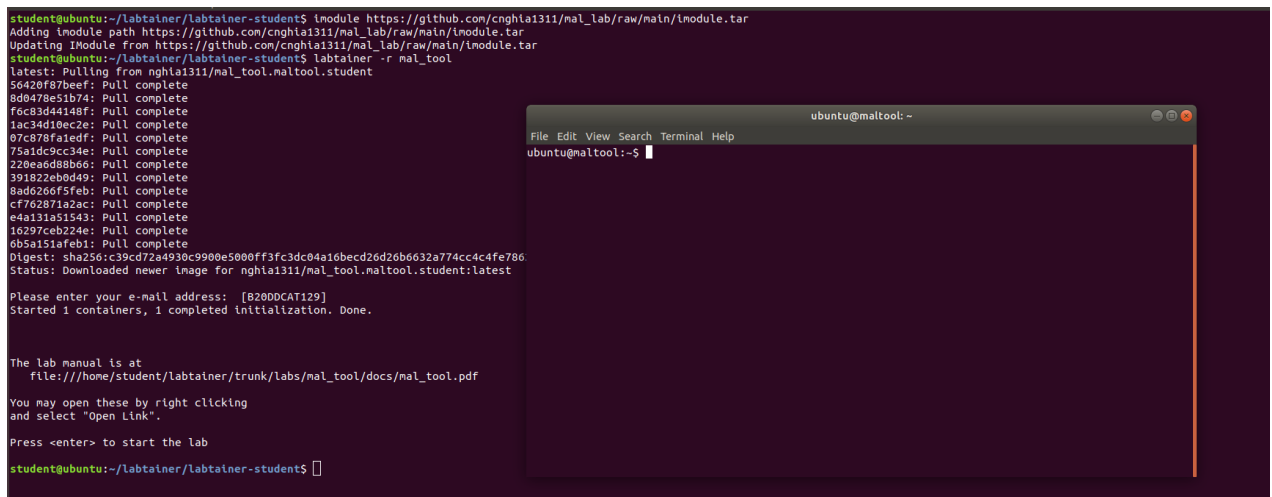


Hình 8 File imodule.tar chứa bài thực hành

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



Hình 9 Đẩy file imodule.tar lên github

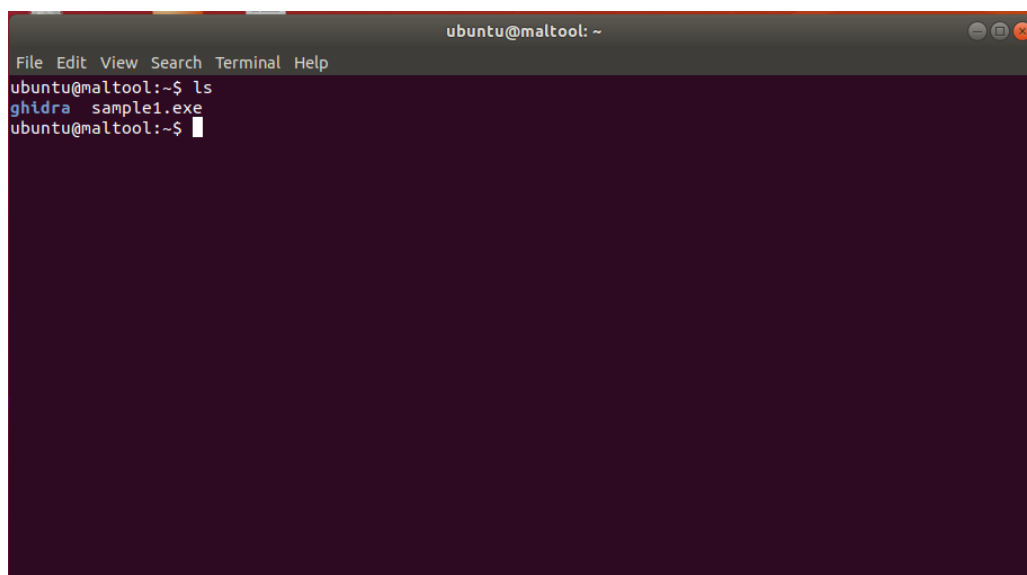


Hình 10 Lấy lab lưu trữ về

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:

- Máy ảo đã có sẵn thư mục ghidra và mẫu mã độc sample1.



Hình 11 Các tệp cần thiết trong máy maltol

- Kiểm tra file thực thi có được nén bằng UPX, sinh viên chạy lệnh:

```

ubuntu@maltool: ~
File Edit View Search Terminal Help
ubuntu@maltool:~$ upx -t sample1.exe
          Ultimate Packer for eXecutables
          Copyright (C) 1996 - 2018
UPX 3.95      Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

testing sample1.exe [OK]

Tested 1 file.

```

Hình 12 Chạy lệnh `upx -t sample1.exe`

- Giải nén tệp thực thi bằng UPX, sinh viên chạy lệnh:

```

ubuntu@maltool:~$ upx -d sample1.exe
          Ultimate Packer for eXecutables
          Copyright (C) 1996 - 2018
UPX 3.95      Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

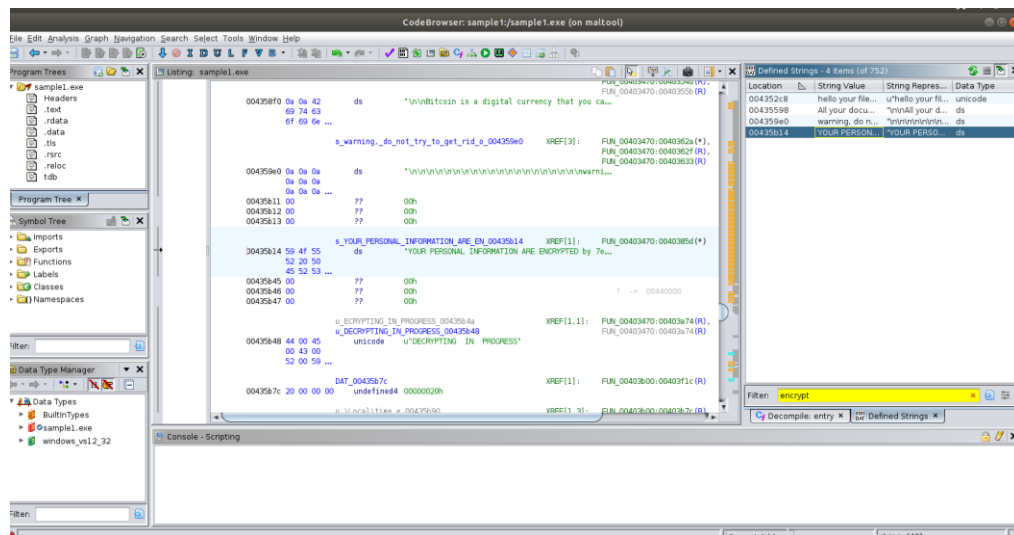
  File size      Ratio      Format      Name
  -----
  322560 <-    128512    39.84%    win32/pe    sample1.exe

Unpacked 1 file.

```

Hình 13 Chạy lệnh `upx -d sample1.exe`

- Phân tích mã nhị phân bằng Ghidra, chuyển đến thư mục Ghidra và khởi chạy:



Hình 14 Tạo thành công project Ghidra và tìm được đầy đủ thông điệp

- Tạo rules và phân loại mã độc bằng Yara, tạo tệp rule Yara dựa trên thông điệp đầy đủ đã ghi nhận.

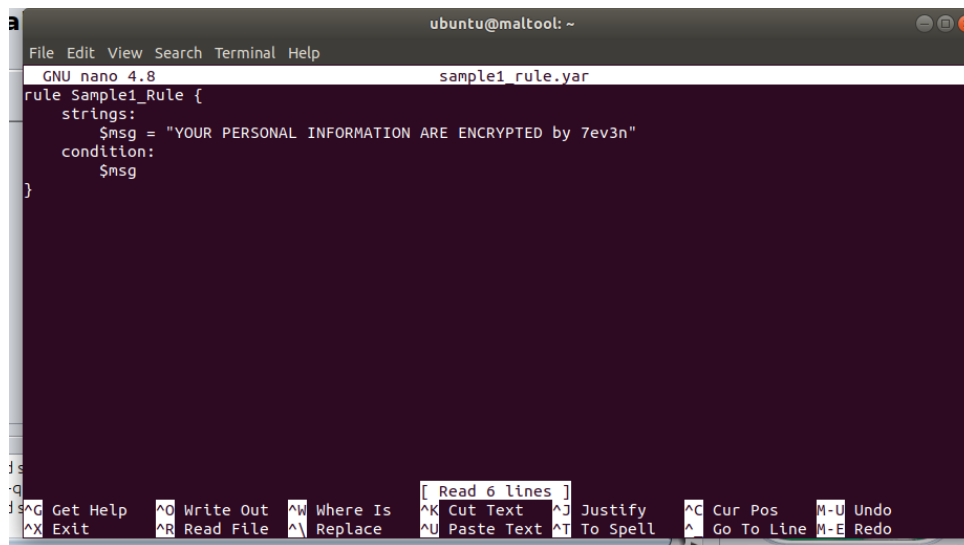
```

ubuntu@maltool:~$ touch sample1_rule.yar
ubuntu@maltool:~$ nano sample1_rule.yar

```

Hình 15 Tạo file rule

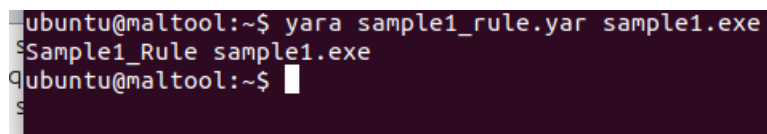
- Dùng lệnh nano để chỉnh viết rule.



```
File Edit View Search Terminal Help
GNU nano 4.8 sample1_rule.yar
rule Sample1_Rule {
  strings:
    $msg = "YOUR PERSONAL INFORMATION ARE ENCRYPTED by 7ev3n"
  condition:
    $msg
}
```

Hình 16 Viết rule

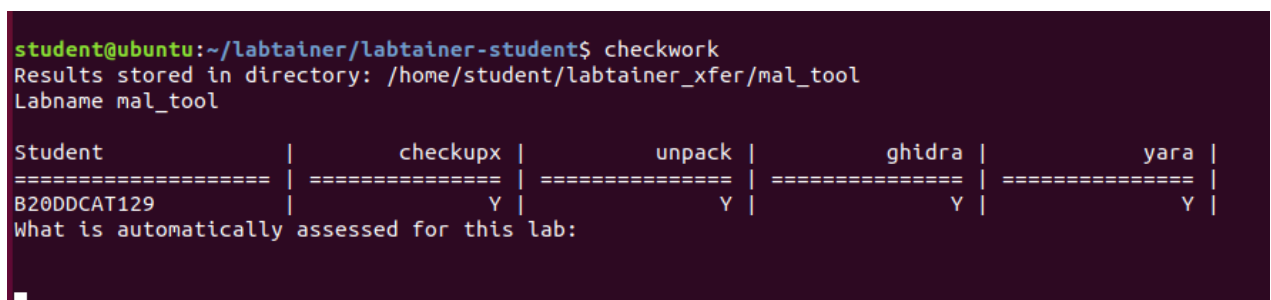
- Lưu tệp rule (ví dụ: sample1_rule.yar) và chạy lệnh:



```
ubuntu@maltool:~$ yara sample1_rule.yar sample1.exe
Sample1_Rule sample1.exe
ubuntu@maltool:~$
```

Hình 17 Chạy lệnh yara sample1_rule.yar sample1.exe

- 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_tool
Labname mal_tool

Student |          checkupx |          unpack |          ghidra |          yara |
===== | ===== | ===== | ===== | ===== |
B20DDCAT129 |          Y |          Y |          Y |          Y |
What is automatically assessed for this lab:
```

Hình 18 Đánh giá kết quả bài thực hành

TÀI LIỆU THAM KHẢO

- [1] M. K. A. Monnappa, Learning Malware Analysis, Birmingham, UK: Packt Publishing, 2018.
- [2] C. Eagle and K. Nance, The Ghidra Book: The Definitive Guide, San Francisco: No Starch Press, 2020.