

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

-----o0o-----



BÁO CÁO BÀI THỰC HÀNH
PHÂN TÍCH TÍNH KỸ THUẬT PERSISTENCE

Giảng viên hướng dẫn: TS. Nguyễn Ngọc Điệp

Sinh viên thực hiện: Phạm Vũ Minh Hiếu – B20DCAT061

Lớp: D20CQAT01-B

Mã sinh viên: B20DCAT061

Hà Nội - 2024

MỤC LỤC

MỤC LỤC	2
DANH MỤC CÁC HÌNH VẼ.....	3
DANH MỤC CÁC BẢNG BIỂU	3
DANH MỤC VIẾT TẮT	4
1.1. Giới thiệu bài thực hành	5
1.2. Nội dung và hướng dẫn bài thực hành	5
1.2.1. Mục đích	5
1.2.2. Yêu cầu đối với sinh viên	5
1.2.3. Nội dung thực hành.....	6
1.3. Phân tích yêu cầu bài thực hành	8
1.4. Thiết kế bài thực hành	9
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	16
1.7. Thử nghiệm và đánh giá	17
TÀI LIỆU THAM KHẢO.....	24

DANH MỤC CÁC HÌNH VẼ

Hình 1. Giao diện Labedit	12
Hình 2. Cài đặt Result 1	13
Hình 3. Cài đặt phần Result 2	13
Hình 4. Cài đặt phần Goals	13
Hình 5. Cài đặt phần Parameter	13
Hình 6. Dockerfiles	14
Hình 7: Add và comit bài lab	14
Hình 8: Thêm bài lab muốn lưu trữ.....	15
Hình 9: Quá trình tải bài lab lên dockerhub.....	15
Hình 10. Đẩy thành công.....	16
Hình 11. Tạo file Imodule.tar.....	16
Hình 12. File imodule.tar chứa bài thực hành.....	16
Hình 13: Đẩy lên github thành công	17
Hình 14: Remote sang máy win 10	17
Hình 15: Thành phần của mal-stic-per.....	18
Hình 16: CFF Explorer.....	18
Hình 17. Giao diện IDA Pro sau khi mở file	19
Hình 18: Hàm main của file độc hại	19
Hình 19: Sơ đồ gọi hàm và API của main	20
Hình 20: Mã chương trình của hàm sub_401040.....	20
Hình 21. Mã hẹn ngày kích hoạt file độc hại	21
Hình 22. Mã chương trình tạo luồng thực thi	21
Hình 23: Mã chương trình của StartAddress	22
Hình 24: Thông tin được lưu vào file ATTT.txt.....	23
Hình 25. Đánh giá kết quả bài thực hành.....	23

DANH MỤC CÁC BẢNG BIỂU

Bảng 1. Bảng Results	10
Bảng 2. Bảng Goals.....	11
Bảng 3. Bảng Paramater.....	12

DANH MỤC VIẾT TẮT

Từ viết tắt	Thuật ngữ tiếng Anh / Giải thích	Thuật ngữ tiếng Việt / Giải thích
API	Application Programming Interface	Giao diện lập trình ứng dụng, cung cấp cách thức cho các chương trình giao tiếp với nhau.
SCP	Secure Copy Protocol	Giao thức sao chép tệp an toàn qua mạng bằng SSH.
DOS	Denial of Service	Tấn công từ chối dịch vụ, làm cho hệ thống hoặc dịch vụ không thể phục vụ người dùng hợp lệ.
DDOS	Distributed Denial of Service	Tấn công từ chối dịch vụ phân tán, sử dụng nhiều nguồn để tấn công một hệ thống, gây quá tải và làm tê liệt.
Per	Persistence	Khả năng mã độc tồn tại trên hệ thống, tự tái khởi động hoặc thực thi ngay cả sau khi khởi động lại.
Mal	Malware	Một file có hành vi độc hại
Stic	Static	Hành phi phân tích tĩnh một file độc hại
Use Xrefs Chart	Use Cross-references Chart	Sử dụng biểu đồ tham chiếu chéo để theo dõi các liên kết hoặc tham chiếu trong mã nguồn hoặc dữ liệu.
	Service	Dịch vụ chạy trong hệ điều hành để cung cấp các chức năng hoặc ứng dụng nền tảng cho người dùng.
	Nerver	Không có định nghĩa rõ ràng, có thể là lỗi chính tả của "Never".
	Offset	Địa chỉ hoặc khoảng cách tương đối từ điểm gốc trong bộ nhớ, tệp hoặc chương trình.

1.1. Giới thiệu bài thực hành

Bài thực hành "**Tìm hiểu về phân tích tĩnh kỹ thuật Persistence**" được thiết kế nhằm giúp sinh viên hiểu rõ hơn về kỹ thuật phân tích tĩnh, đặc biệt là cách mã độc áp dụng kỹ thuật **persistence** để duy trì sự tồn tại trên hệ thống. Đây là một nội dung quan trọng giúp sinh viên nhận thức được cách thức hoạt động của mã độc trong việc tạo ra các cơ chế khởi chạy tự động hoặc ẩn mình trên hệ thống Windows.

Trong bài thực hành này, sinh viên sẽ sử dụng **IDA Pro**, một công cụ phân tích mã tĩnh mạnh mẽ, để tìm hiểu cách mã độc triển khai kỹ thuật persistence. Bằng cách quan sát các chuỗi, lời gọi hàm, và cấu trúc mã, sinh viên sẽ phân tích được cách mã độc tương tác với registry, dịch vụ khởi động hoặc các cơ chế hệ thống khác để đảm bảo khả năng tự động khởi chạy khi hệ thống được bật.

Mục tiêu của bài thực hành là giúp sinh viên:

- Làm quen với việc sử dụng **IDA Pro** để phân tích tĩnh mã độc.
- Xác định các kỹ thuật persistence phổ biến trong mã độc Windows.
- Nhận diện và hiểu được cách thức mã độc tương tác với các thành phần hệ thống nhằm duy trì sự tồn tại.

Thông qua bài thực hành, sinh viên sẽ nắm vững quy trình phân tích tĩnh cơ bản với **IDA Pro** và hiểu rõ hơn về các kỹ thuật persistence. Đây là một bước quan trọng trong việc xây dựng nền tảng kỹ năng phân tích mã độc, chuẩn bị cho các tình huống thực tế liên quan đến bảo mật hệ thống và xử lý mã độc trong tương lai.

1.2. Nội dung và hướng dẫn bài thực hành

1.2.1. Mục đích

Giúp sinh viên tìm hiểu khái niệm về phân tích mã độc và sử dụng công cụ **IDA Pro** để phân tích kỹ thuật persistence trên mã độc Windows. Qua đó, sinh viên sẽ làm quen với cách phân tích tĩnh, nhận diện các hành vi đáng ngờ (như chỉnh sửa registry, tạo dịch vụ tự động khởi chạy, thao tác với task scheduler) và phát hiện các dấu hiệu persistence được mã độc triển khai.

1.2.2. Yêu cầu đối với sinh viên

- Có kiến thức cơ bản về hệ điều hành Windows, đặc biệt là cách hệ thống khởi động và cách hoạt động của các thành phần như registry, dịch vụ, và task scheduler.
- Đã làm quen với công cụ **IDA Pro** và biết cách đọc mã assembly cơ bản.
- Hiểu các khái niệm cơ bản về phân tích tĩnh mã độc, đặc biệt là kỹ thuật **persistence**.

1.2.3. Nội dung thực hành

Khởi động bài lab:

Vào terminal, gõ:

labtainer -r mal-stic-per

(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 sẽ có 1 terminal ảo sẽ xuất hiện, có tên là mal-stic. Trên terminal **mal-stic-per** đã được cài đặt sẵn công cụ để remote tới máy Windows chứa mã độc là Xfreerdp.

xfreerdp /u:user /p:password /v:ip

(user: tài khoản máy Window, password: mật khẩu của tài khoản, ip: IP của máy Window)

Ví dụ: xfreerdp /v:192.168.18.128 /u:minhh /p:1

Sau khi remote thành công sang máy Window, sẽ thấy 1 folder **mal-stic-per** trên màn hình, bên trong đó có sẵn 1 file mã độc và 1 file ATTT.txt (điền output mà bài yêu cầu để checkwork)

- Sinh viên sử dụng công cụ CFF để kiểm tra định dạng chuẩn của file và kéo vào phiên bản IDA Pro nào cho hợp lý
- Sinh viên mở IDA Pro lên và kéo thả file thực thi cho sẵn vào để bắt đầu đánh giá, kiểm tra file đã cho.

- Sinh viên ở ngay hàm **main** đọc đoạn mã, và cho biết file độc hại làm như thế nào để nó có thể tiếp tục chạy khi máy tính khởi động lại. Viết tên phương thức và tên của nó thu thập được trong code vào file ATTT.txt

Service

MalService

- Sinh viên sử dụng Use xrefs chart để xem tại độ sâu 1 thì hàm **main** gọi trực tiếp các API nào, và cho biết API đấy có tác dụng gì. Viết vào file ATTT.txt tên API tìm được.

StartServiceCtrlDispatcherA

- Sinh viên tiến vào trong hàm **sub_401040** để đọc code, và giải thích tại sao chương trình lại dùng mutex, và điền vào file ATTT.txt mutex đấy là gì.

HGL345

- Sinh viên đọc đoạn code trong hàm **sub_401040** và xác định xem vào ngày nào mã độc sẽ tạo ra các luồng mới để thực hiện hành vi độc hại. Điền vào file ATTT.txt ngày mà file độc hại bắt đầu thực hiện hành vi của mình và số luồng mà file độc hại sẽ tạo ra khi bắt đầu chạy.

1/1/2100

20

- Sinh viên đi xuống cuối của hàm **sub_401040** và tìm tới offset chứa các đường dẫn và công cụ kết nối tới đường dẫn đó, kết hợp với thông tin bên trên xác định xem mục đích của file độc hại là gì. Viết vào file ATTT.txt thông tin về đường dẫn tìm được và công cụ kết nối tới đường dẫn đấy.

<http://www.malwareanalysisbook.com>

Internet Explorer 8.0

- Sau khi xong hết các nhiệm vụ, sinh viên lưu lại file.
- Về lại terminal mal-stic dùng scp để tiến hành kéo thư mục **mal-stic-per** từ máy Windows về (có thể dùng cách khác để kéo file về mà sinh viên biết).

scp -r minh@192.168.18.128:"C:/Users/minh/Desktop/mal-stic-per" ~/

- Lệnh sẽ tự tạo 1 folder share trên máy Linux, tiến hành cd sang folder **mal-stic** và cat file ATTT.txt

cd mal-stic-per

cat ATTT.txt

- Cuối cùng ta sẽ checkwork
- Kết thúc bài lab:

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

stoplab mal-stic-per

Khi bài lab kết thúc, một tệp zip lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab.

Khởi động lại bài lab:

Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

startlab -r mal-stic-per

1.3. Phân tích yêu cầu bài thực hành

Bài thực hành yêu cầu sinh viên làm quen với công cụ **IDA Pro** để phân tích kỹ thuật **persistence** được sử dụng trong mã độc Windows. Kỹ thuật **persistence** là một phương thức mà mã độc sử dụng để duy trì sự tồn tại trên hệ thống, ngay cả khi hệ thống khởi động lại. Trong bài thực hành này, sinh viên sẽ phân tích một file mã độc được cung cấp sẵn trong môi trường làm việc để xác định các kỹ thuật **persistence**, thông qua việc kiểm tra các đoạn mã nguồn, API được gọi, và các cấu trúc hệ thống mà mã độc can thiệp.

Sinh viên sẽ bắt đầu bài thực hành bằng lệnh labtainer <tên bài lab> và nhập MSV của mình để khởi tạo môi trường làm việc. Sau đó, sinh viên sử dụng **IDA Pro** để mở file mã độc và phân tích các đoạn mã chính, đặc biệt là hàm main và các hàm con liên quan. Sinh viên cần xác định cách mã độc triển khai các kỹ thuật **persistence**, như tạo dịch vụ tự động, chỉnh sửa registry, hoặc sử dụng các cơ chế khác để duy trì khả năng tự khởi động khi hệ thống

bật lại. Thông tin thu thập được bao gồm tên dịch vụ, các API được gọi, và các đoạn mã xác định hành vi mã độc sẽ được ghi vào file **ATTT.txt**.

Khi hoàn thành bài thực hành, sinh viên tải file **ATTT.txt** từ máy Windows về máy Linux bằng lệnh `scp` và dừng bài lab bằng lệnh `stoplab <tên bài lab>`. File kết quả phải bao gồm các thông tin về kỹ thuật **persistence** được mã độc sử dụng, API và hàm liên quan, cũng như các thông tin cần thiết khác như mutex, số luồng được tạo ra, hoặc ngày mã độc kích hoạt hành vi độc hại. Đây là bài thực hành cơ bản giúp sinh viên hiểu rõ kỹ thuật **persistence** trong mã độc và cách phân tích tĩnh mã độc bằng **IDA Pro**.

1.4. Thiết kế bài thực hành

Trên môi trường máy ảo Ubuntu được cung cấp, sử dụng docker tạo ra container mang tên “mal-stic-per”.

Cấu hình docker gồm có:

- mal-stic: Lưu cấu hình cho máy thực hành, trong đó gồm có:
 - Tên máy: mal-stic-per
- Config: Lưu cấu hình hoạt động của hệ thống.
- Dockerfile: Mô tả cấu hình của container mal-stic-per, trong đó:
 - mal-stic: Sử dụng các thư viện mặc định của hệ thống và tích hợp sẵn công cụ xfreerdp và dcp
 - xfreerdp: Hỗ trợ remote sang máy tính Windows
 - scp: Dùng để truyền file từ máy Windows về Linux

Trên môi trường máy Windows được cài:

- Mở tính năng remote cho phép máy ubuntu remote tới
- SSH server
- Thư mục gồm file độc hại để phân tích và một file để điền kết quả.

- Các công cụ dùng để phân tích như CFF Explorer, IDA Pro.

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

- Phân tích kỹ thuật persistence trong hàm main
- Phân tích các API được gọi bởi hàm main.
- Phân tích mutex trong hàm sub_401040.
- Xác định thời gian kích hoạt/kết thúc và số luồng.
- Phân tích các đường dẫn độc hại và công cụ kết nối.

Kết quả của từng nhiệm vụ sẽ được ghi lại trong file ATTT.txt, sau đó chuyển qua hệ thống Labtainer.

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 1,2,3:

Bảng 1. Bảng Results

Result Tag	Container	File	Field Type	Field ID	Timestamp Type
_where-per	mal-stic-per	cat.stdout	CONTAINS	Service	File
_name-service	mal-stic-per	cat.stdout	CONTAINS	MalService	File
mutex	mal-stic-per	cat.stdout	CONTAINS	HGL345	File

network	mal-stic-per	cat.stdout	CONTAINS	http://www.malwareanalysisbook.com	File
time-start	mal-stic-per	cat.stdout	CONTAINS	1/1/2100	File
thread-ddos	mal-stic-per	cat.stdout	CONTAINS	20	File
time-end	mal-stic-per	cat.stdout	CONTAINS	Never	File
api-main	mal-stic-per	cat.stdout	CONTAINS	StartServiceCtrlDispatcherA	File
tool-ddos	mal-stic-per	cat.stdout	CONTAINS	Internet Explorer 8.0	File

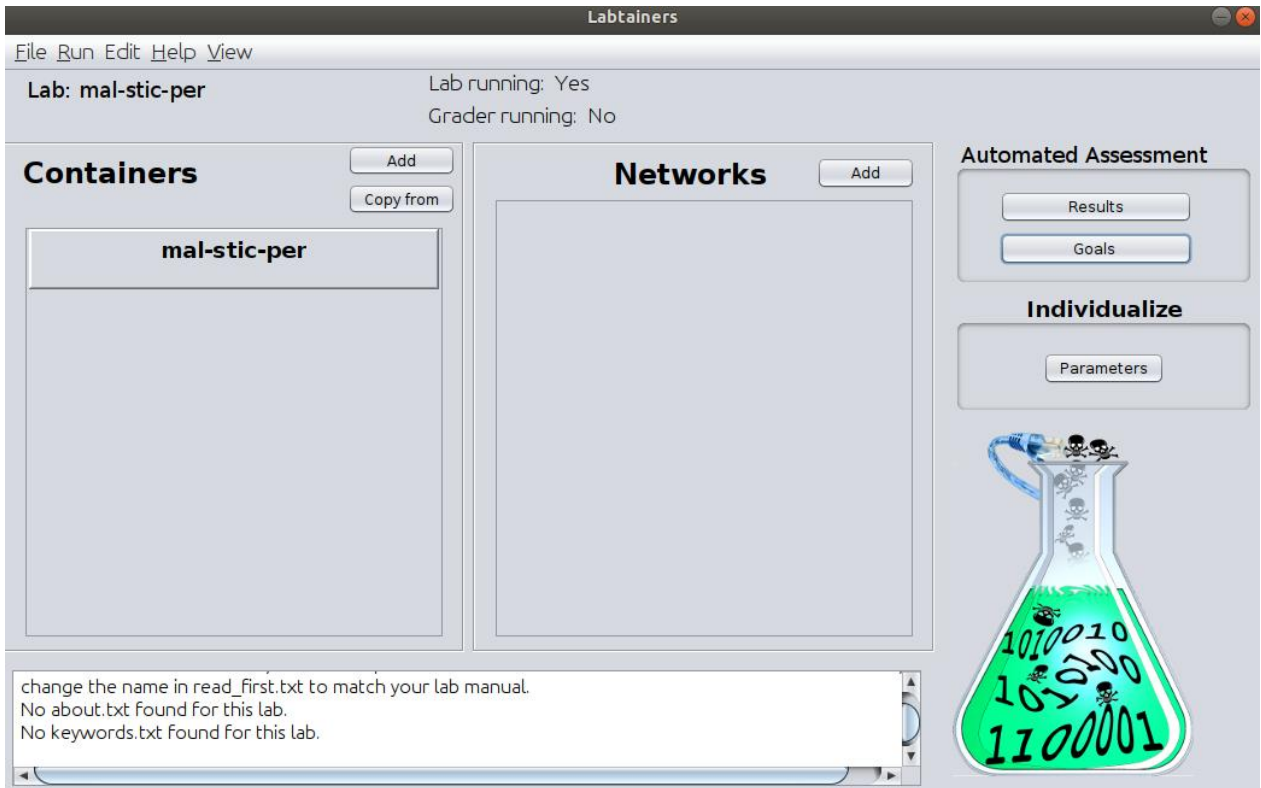
Bảng 2. Bảng Goals

Goal ID	Goal Type	Boolean	Boolean Result Tags
name-per	boolean	_where-per and _name-service	_where-per

Bảng 3. Bảng Paramater

Param ID	Operator	File name	Symbol	Step	Hashstring

1.5. Cài đặt và cấu hình các máy ảo



Hình 1. Giao diện Labedit

The screenshot shows a window titled "Goals for mal-stic-per" with "Create" and "Remove All" buttons. It contains a table with four rows of configured result tags. Each row has columns for Result Tag, Container, File, Field Type, Field ID, and Timestamp Type. The first row is selected.

	Result Tag	Container	File	Field Type	Field ID	Timestamp Type
1	_where-per	mal-stic-per	cat.stdout	CONTAINS	Service	File
2	_name-service	mal-stic-per	cat.stdout	CONTAINS	MalService	File
3	mutex	mal-stic-per	cat.stdout	CONTAINS	HGL345	File
4	network	mal-stic-per	cat.stdout	CONTAINS	areanalysisbook.com	File

Hình 2. Cài đặt Result 1

The screenshot shows the same window with five rows of configured result tags. The first row is selected.

	Result Tag	Container	File	Field Type	Field ID	Timestamp Type
5	time-start	mal-stic-per	cat.stdout	CONTAINS	1/1/2100	File
6	thread-ddos	mal-stic-per	cat.stdout	CONTAINS	20	File
7	time-end	mal-stic-per	cat.stdout	CONTAINS	Never	File
8	api-main	mal-stic-per	cat.stdout	CONTAINS	serviceCtrlDispatcherA	File
9	tool-ddos	mal-stic-per	cat.stdout	CONTAINS	Internet Explorer 8.0	File

Hình 3. Cài đặt phần Result 2

The screenshot shows the same window with one configured goal. The goal is named "name-per" and is of type "boolean".

	Goal ID	Goal Type	Boolean	Boolean Result Tags
1	name-per	boolean	_where-per and _name-service	_where-per

Hình 4. Cài đặt phần Goals

The screenshot shows a window titled "Parameters (Individualize) for mal-stic-per" with "Create" and "Remove All" buttons. It contains a table with columns for Param ID and Operator, but it is currently empty.

Param ID	Operator
----------	----------

Hình 5. Cài đặt phần Parameter



```
#
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

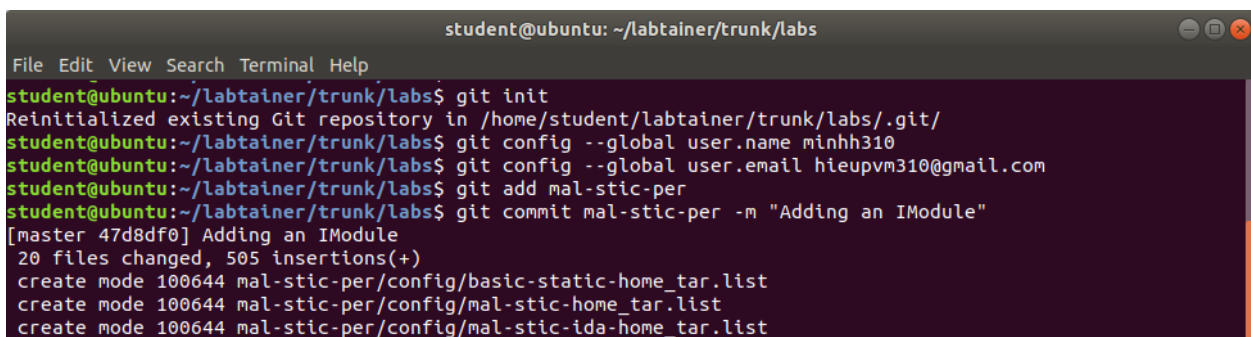
RUN apt-get update
RUN apt install freerdp2-x11 -y
#
# put package installation here, e.g.,
#   RUN apt-get update && apt-get install -y --no-install-recommends somepackage
#
#
# Install the system files found in the _system directory
#
ADD $labdir/$imagedir/sys_tar/sys.tar /
ADD $labdir/sys_$lab.tar.gz /
#
RUN useradd -ms /bin/bash $user_name
RUN echo "$user_name:$password" | chpasswd
RUN adduser $user_name sudo
# replace above with below for centos/fedora
#RUN usermod $user_name -a -G wheel

#
# **** Perform all root operations, e.g., ****
# **** "apt-get install" prior to the USER command. ****
#
USER $user_name
ENV HOME /home/$user_name
```

Hình 6. Dockerfiles

1.6. Tích hợp và triển khai

1.6.1. Docker Hub



```
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 config --global user.name minh310
student@ubuntu:~/labtainer/trunk/labs$ git config --global user.email hieupvm310@gmail.com
student@ubuntu:~/labtainer/trunk/labs$ git add mal-stic-per
student@ubuntu:~/labtainer/trunk/labs$ git commit mal-stic-per -m "Adding an IModule"
[master 47d8df0] Adding an IModule
20 files changed, 505 insertions(+)
create mode 100644 mal-stic-per/config/basic-static-home_tar.list
create mode 100644 mal-stic-per/config/mal-stic-home_tar.list
create mode 100644 mal-stic-per/config/mal-stic-ida-home_tar.list
```

Hình 7: Add và comit bài lab

```

student@ubuntu:~/labtainer/trunk/labs$ git commit mal-stic-per -m "Adding an IModule"
[master 47d8df0] Adding an IModule
20 files changed, 505 insertions(+)
create mode 100644 mal-stic-per/config/basic-static-home_tar.list
create mode 100644 mal-stic-per/config/mal-stic-home_tar.list
create mode 100644 mal-stic-per/config/mal-stic-ida-home_tar.list
create mode 100644 mal-stic-per/config/mal-stic-per-home_tar.list
create mode 100644 mal-stic-per/config/parameter.config
create mode 100644 mal-stic-per/config/start.config
create mode 100644 mal-stic-per/dockerfiles/Dockerfile.mal-stic-per.mal-stic-per.student
create mode 100644 mal-stic-per/docs/read_first.txt
create mode 100644 mal-stic-per/instr_config/goals.config
create mode 100755 mal-stic-per/instr_config/pregrade.sh
create mode 100644 mal-stic-per/instr_config/results.config
create mode 100644 mal-stic-per/mal-stic-per/_bin/.treataslocal.swo
create mode 100755 mal-stic-per/mal-stic-per/_bin/fixlocal.sh
create mode 100644 mal-stic-per/mal-stic-per/_bin/treataslocal
create mode 100644 mal-stic-per/mal-stic-per/_system/etc/login.defs
create mode 100644 mal-stic-per/mal-stic-per/_system/etc/securetty
create mode 100644 mal-stic-per/mal-stic-per/basic-static.basic-static.student.tar.gz
create mode 100644 mal-stic-per/mal-stic-per/home_tar/home.tar
create mode 100644 mal-stic-per/mal-stic-per/sys_basic-static.basic-static.student.tar.gz
create mode 100644 mal-stic-per/mal-stic-per/sys_tar/sys.tar

```

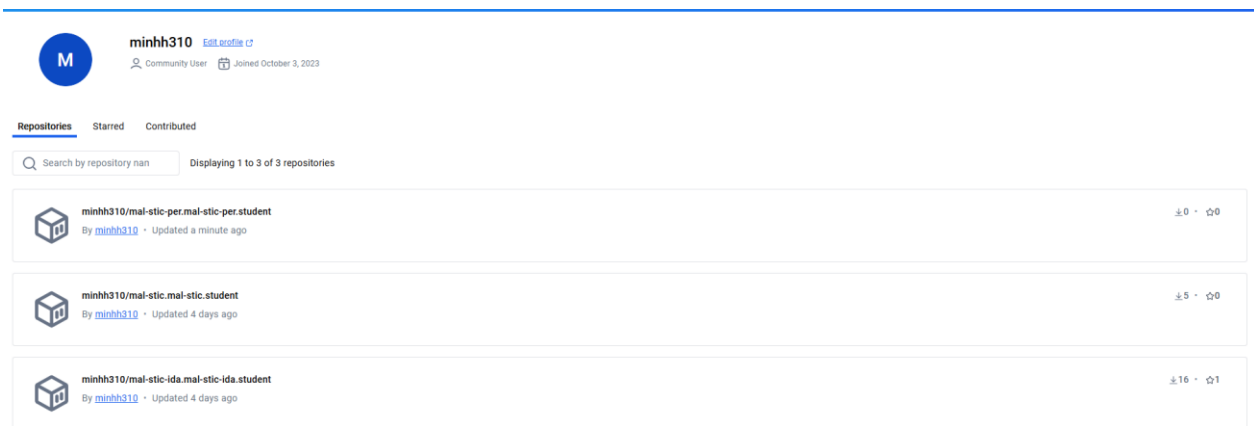
Hình 8: Thêm bài lab muốn lưu trữ

```

student@ubuntu:~/labtainer/trunk/distrib$ ./publish.py -d -l mal-stic-per
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]

```

Hình 9: Quá trình tải bài lab lên dockerhub



Hình 10. Đẩy thành công

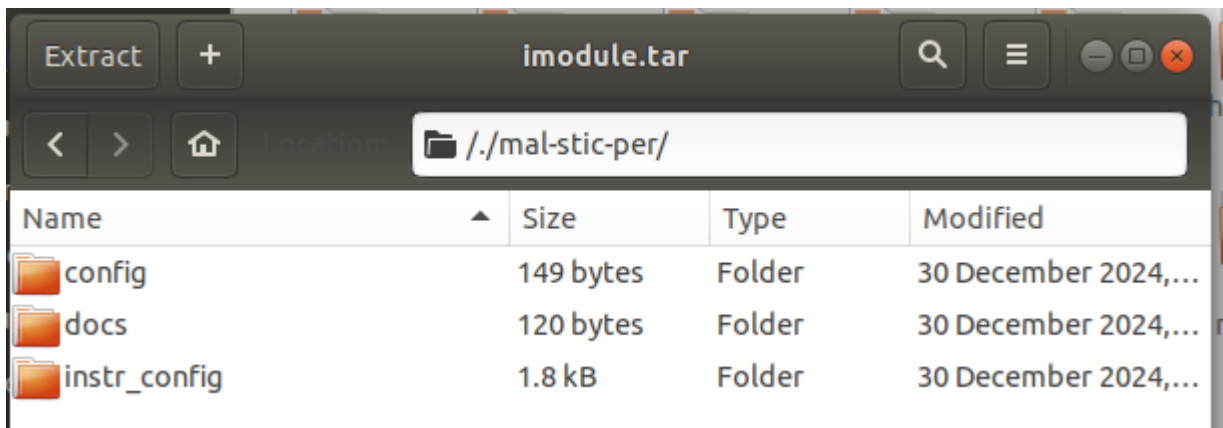
1.6.2. Github

```

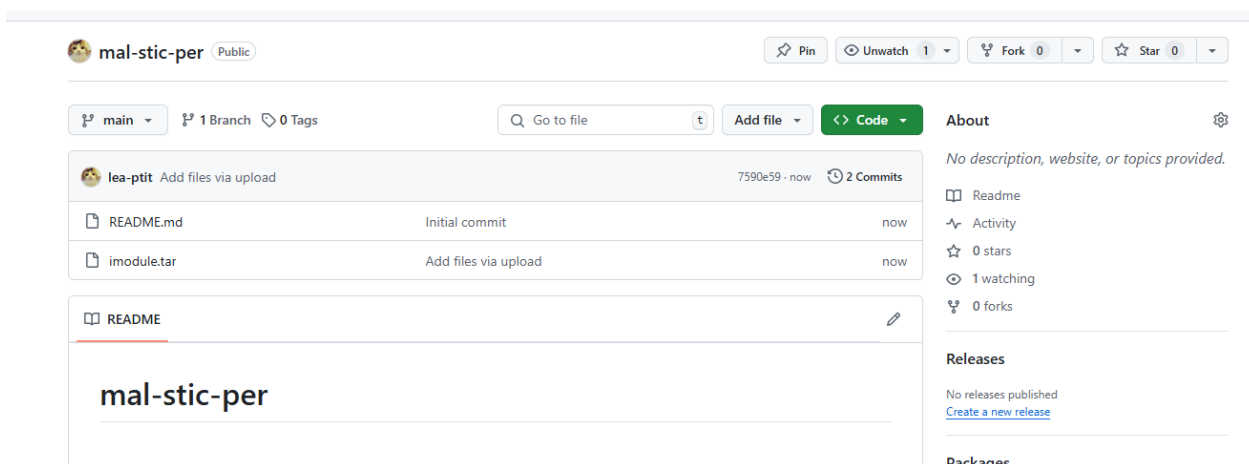
student@ubuntu: ~/labtainer/trunk/distrib
File Edit View Search Terminal Help
student@ubuntu:~/labtainer/trunk/distrib$ create-imodules.sh
lab is mal-stic-per
Do docs
*****
** Post /home/student/labtainer/trunk/imodule.tar to your web server **
*****
student@ubuntu:~/labtainer/trunk/distrib$

```

Hình 11. Tạo file Imodule.tar



Hình 12. File imodule.tar chứa bài thực hành

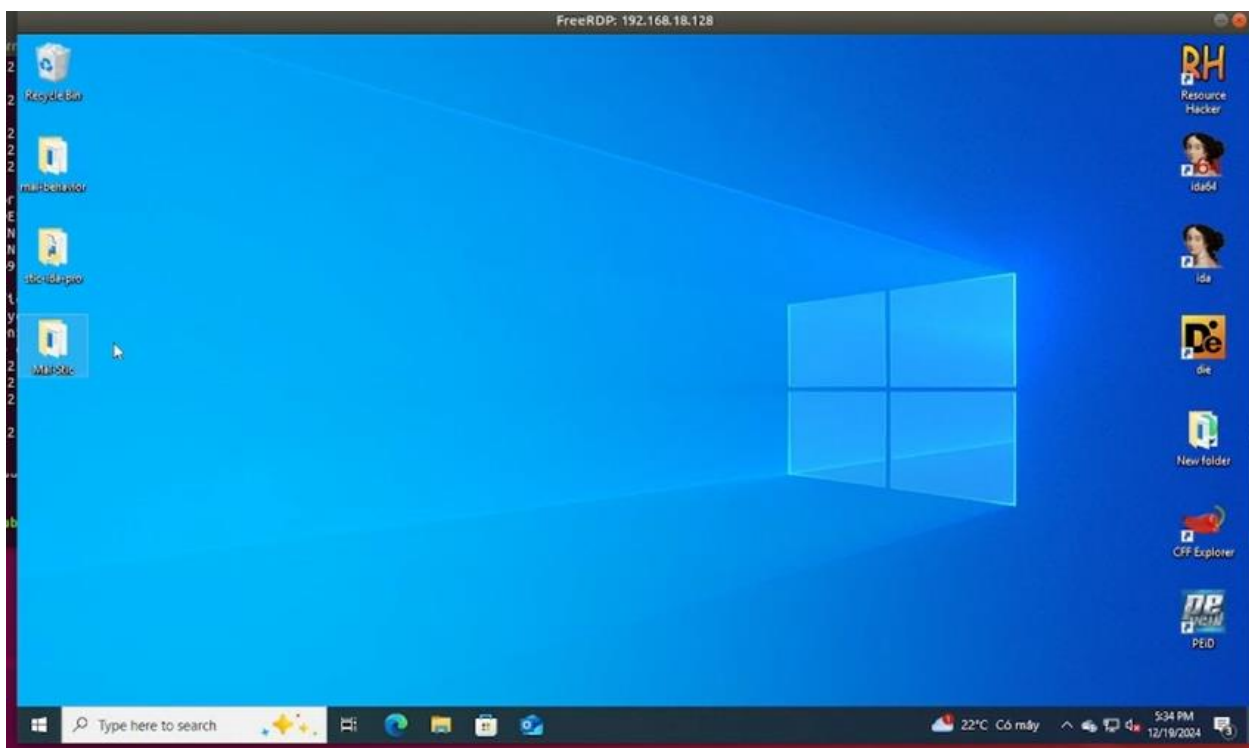


Hình 13: Đẩy lên github thành công

1.7. Thử nghiệm và đánh giá

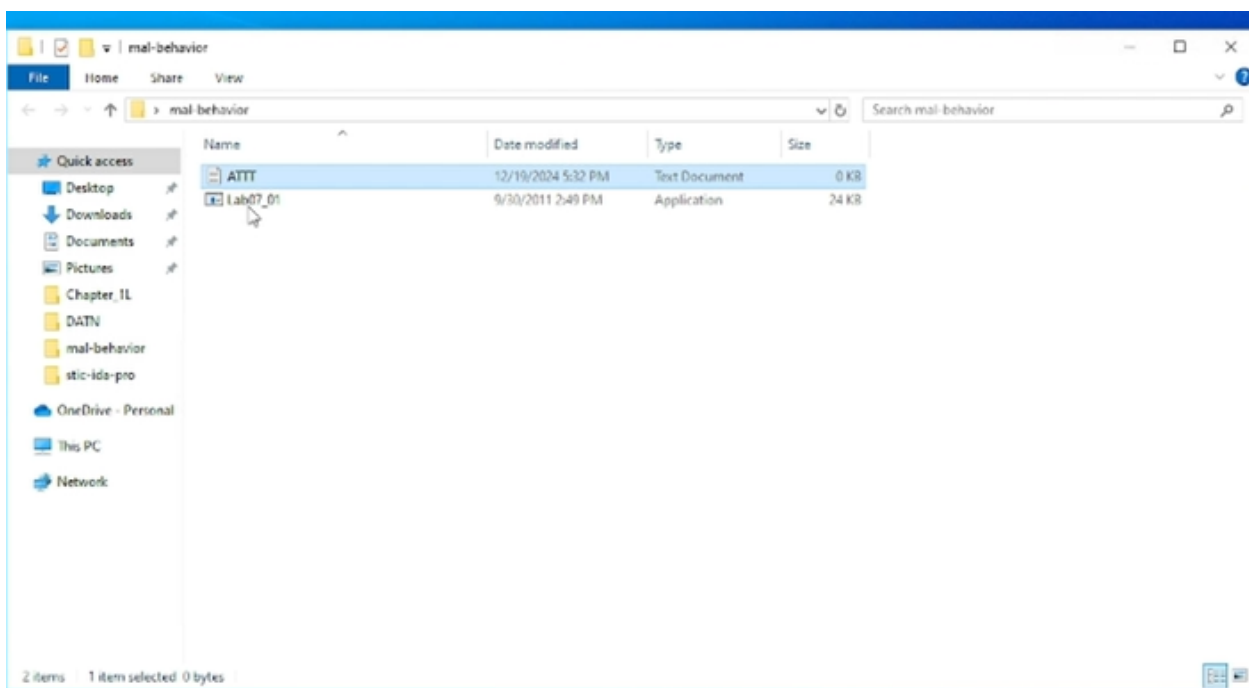
Bài thực hành đã được xây dựng thành công, dưới đây là hình ảnh minh họa về bài thực hành:

Khi bắt đầu labtainer mal-stic-per, ta tiến hành remote tới máy Windows 10 để thao tác và phân tích tĩnh để giải quyết các nhiệm vụ.



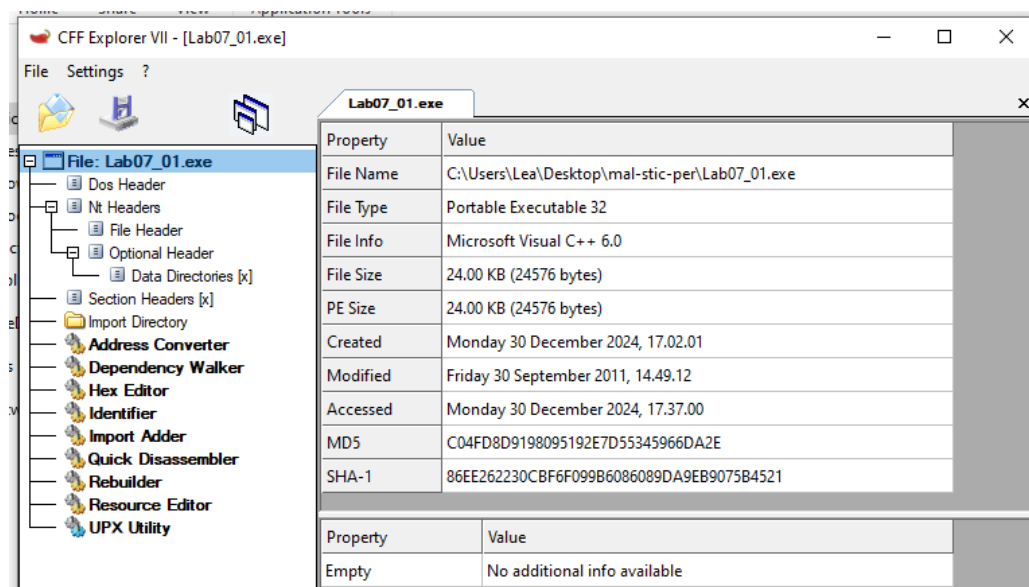
Hình 14: Remote sang máy win 10

Sau khi remote tới được máy chủ Windows 10 thì sinh viên mở thư mục **mal-stic-per** lên để kiểm tra xem đủ 3 file cần thiết hay chưa. Gồm có ATTT.txt, Lab07-1.dll.



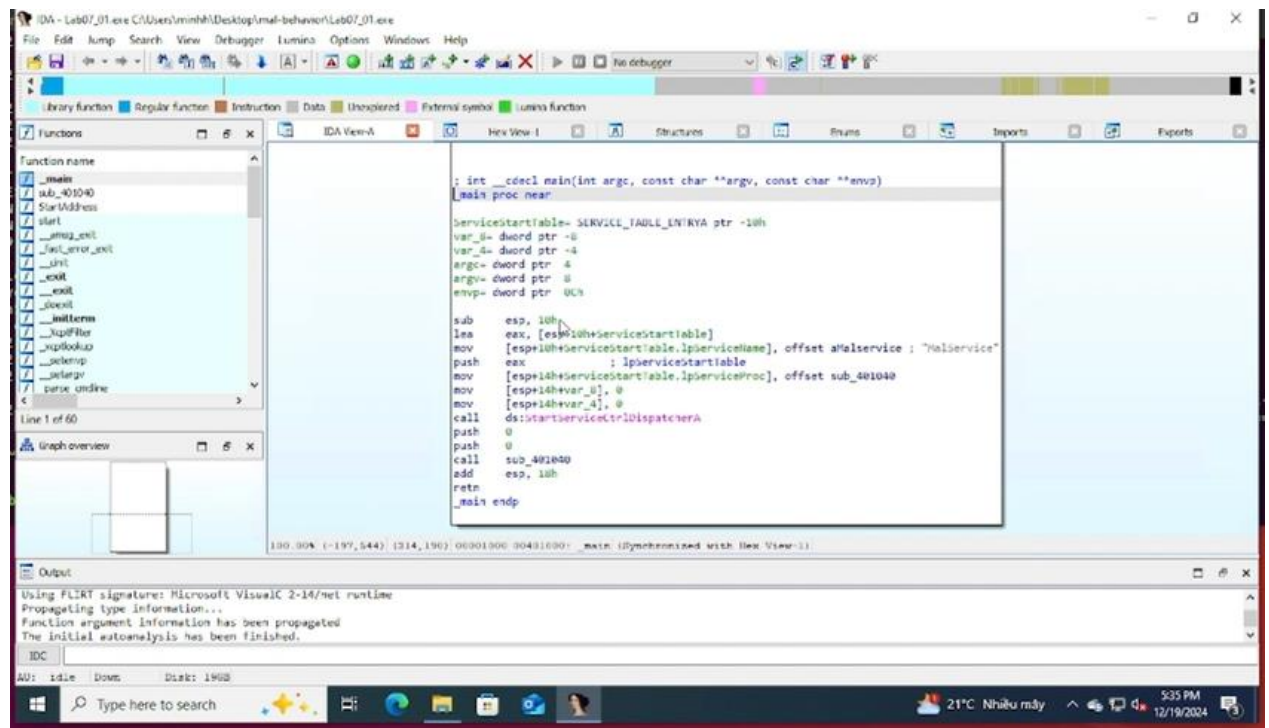
Hình 15: Thành phần của mal-stic-per

Sinh viên kéo thả file vào công cụ CFF Explorer để xác định định dạng file để chọn IDA 32 hay IDA 64 cho phù hợp



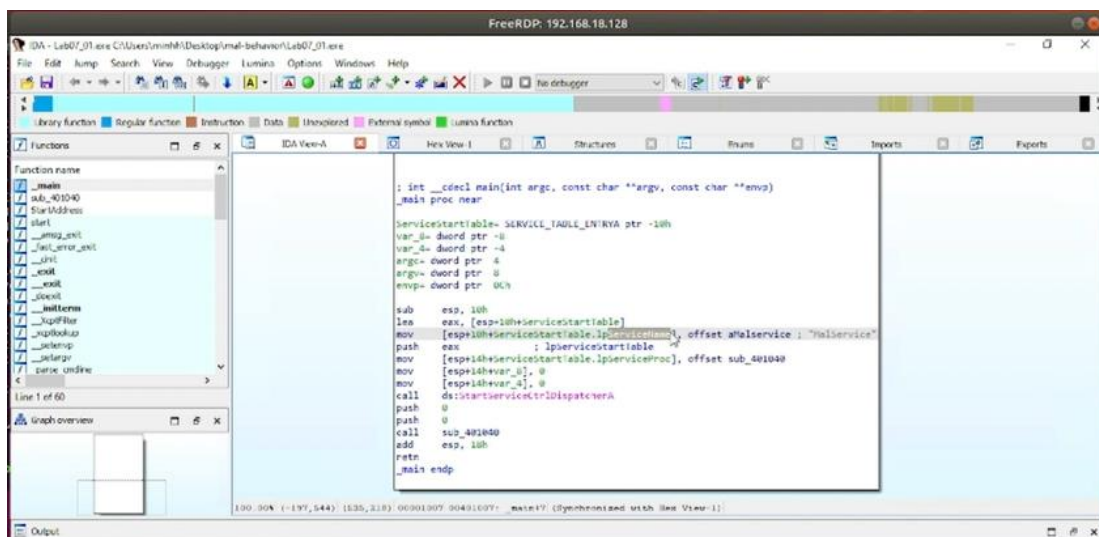
Hình 16: CFF Explorer

Sau khi xác định được định dạng file thì ta mở file đã cho với công cụ IDA Pro mà đã xác định. Sau khi mở thì sẽ hiển thị như bên dưới.



Hình 17. Giao diện IDA Pro sau khi mở file

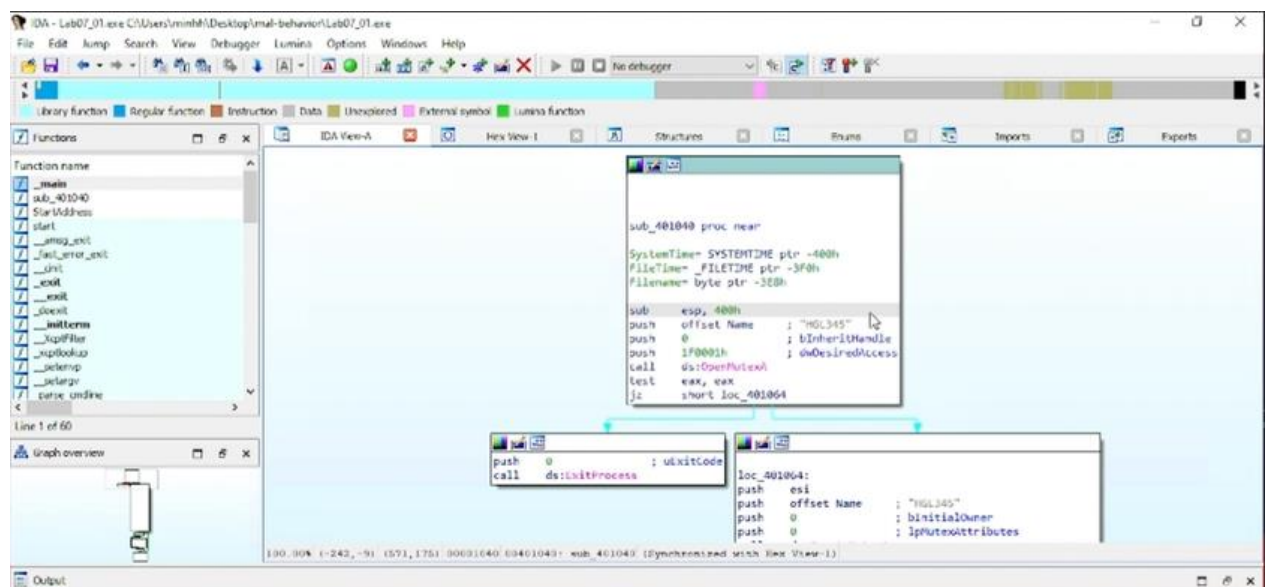
Ở ngay giao diện code của hàm đầu tiên đọc các API và các biến để biết được file độc hại dùng cách gì để kiên trì tồn tại trong máy tính nạn nhân.



Hình 18: Hàm main của file độc hại

The screenshot displays the WinGraph32 application window titled "WinGraph32 - User defined xrefs chart". The main area shows a control flow graph (CFG) for the function `_main`. The graph starts with a blue box labeled `_main`, which branches into two paths: one leading to a pink box labeled `StartCaretakerOnDispatcher` and another leading to a black box labeled `sub_401040`. Below `sub_401040`, there is an ellipsis (`...`) indicating further code. The left sidebar contains a "Functions" list with various symbols like `_main`, `sub_401040`, `Start`, etc. Below the functions list is a "Graph overview" section showing a small thumbnail of the graph. At the bottom, the "Output" window displays text: "Using FLIAT signature: Microsoft VisualC 2-14/net", "Propagating type information...", "Function argument information has been propagated.", and "The initial xrefanalysis has been finished." The bottom status bar shows "AU: idle", "Down", and "Disk: 1908". The Windows taskbar at the very bottom includes the search bar and several application icons.

Sau đó ta vào hàm **sub_401040** để kiểm tra nội dung tiếp theo. Và tìm hiểu xem tại sao file độc hại lại dùng mutex.



20

Sau khi hoàn thành nhiệm vụ về mutex, lướt xuống phía dưới ta có thể thấy được hàm API WaitForSingleObject, và một đoạn mã bên trên dùng để hẹn lịch kích hoạt cho file độc hại vào ngày 1/1/2100, và không thấy kết thúc vào lúc nào.

```

push     esi                ; hSCManager
call     ds:CreateServiceA
xor      edx, edx
lea      eax, [esp+404h+FileTime]
mov      dword ptr [esp+404h+SystemTime.wYear], edx
lea      ecx, [esp+404h+SystemTime]
mov      dword ptr [esp+404h+SystemTime.wDayOfWeek], edx
push     eax                ; lpFileTime
mov      dword ptr [esp+408h+SystemTime.wHour], edx
push     ecx                ; lpSystemTime
mov      dword ptr [esp+40Ch+SystemTime.wSecond], edx
mov      [esp+40Ch+SystemTime.wYear], 834h
call     ds:SystemTimeToFileTime
push     0                  ; lpTimerName
push     0                  ; bManualReset
push     0                  ; lpTimerAttributes

```

Hình 21. Mã hẹn ngày kích hoạt file độc hại

Lướt xuống đoạn code bên dưới, ta thấy được hàm CreateThread dùng để tạo các luồng mới cho file thực thi, và ta thấy được tham số truyền vào là 20. Từ đó biết rằng khi được kích hoạt thì chương trình sẽ tạo ra 20 luồng để làm nhiệm vụ của nó.

```

push     edi
mov      edi, ds:CreateThread
mov      esi, 20

loc_401126:                ; lpThreadId
push     0
push     0                  ; dwCreationFlags
push     0                  ; lpParameter
push     offset StartAddress ; lpStartAddress
push     0                  ; dwStackSize
push     0                  ; lpThreadAttributes
call     edi ; CreateThread
dec      esi
jnz      short loc_401126

```

Hình 22. Mã chương trình tạo luồng thực thi

Lướt xuống dưới ta thấy một offset là StartAddress, kiểm tra thì ta thấy được rằng nhiệm vụ của nó là trở yêu cầu tới URL với dwFlags là 80000000h qua công cụ Internet Explorer 8.0.

```
; Attributes: noreturn
; DWORD __stdcall StartAddress(LPVOID lpThreadParameter)
StartAddress proc near
    lpThreadParameter= dword ptr 4

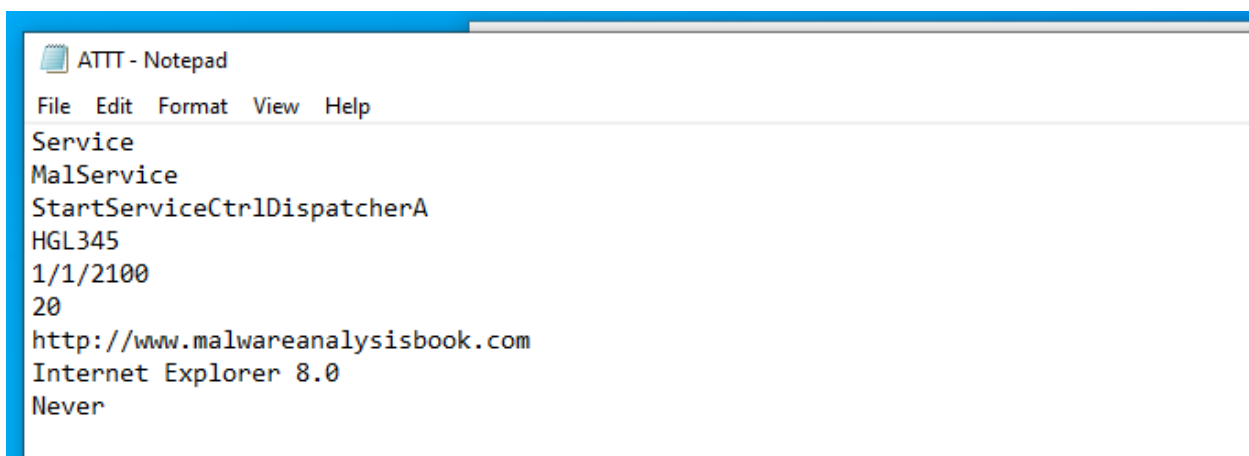
    push    esi
    push    edi
    push    0                ; dwFlags
    push    0                ; lpzProxyBypass
    push    0                ; lpzProxy
    push    1                ; dwAccessType
    push    offset szAgent    ; "Internet Explorer 8.0"
    call    ds:InternetOpenA
    mov     edi, ds:InternetOpenUrlA
    mov     esi, eax
```

```
loc_40116D:                ; dwContext
push    0
push    80000000h          ; dwFlags
push    0                  ; dwHeadersLength
push    0                  ; lpzHeaders
push    offset szUrl        ; "http://www.malwareanalysisbook.com"
push    esi                ; hInternet
call    edi ; InternetOpenUrlA
jmp     short loc_40116D
StartAddress endp
```

Hình 23: Mã chương trình của StartAddress

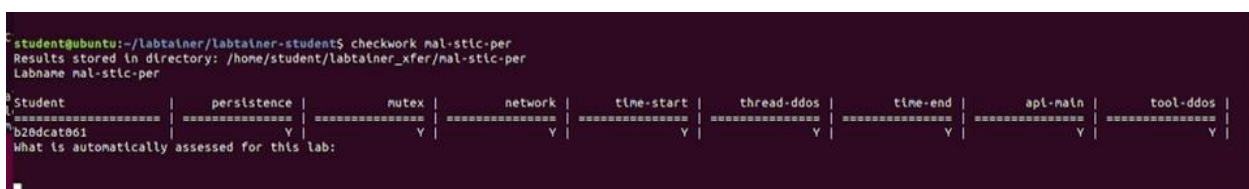
Từ những thông tin đã phân tích được bên trên ta có thể biết rằng file độc hại này dùng để Ddos URL là **<http://www.malwareanalysisbook.com>**, nó được hẹn ngày kích hoạt vào lúc 00h00 ngày 1/1/2100, ngay khi vừa được kích hoạt thì nó sẽ tạo ra 20 luồng liên tục gửi các request tới URL đó một cách vô hạn nhằm mục đích đánh sập, vô hiệu hóa dịch vụ của trang web.

Sau khi làm xong thì ta lưu thông tin lại vào trong file ATTT.txt để kéo về linux và cat file để check lại kết quả lần cuối.



Hình 24: Thông tin được lưu vào file ATTT.txt

Sau khi kéo được file về và cat để check kết quả lần cuối thì ta thu được checkwork đã hoàn thiện bài làm như sau.



Hình 25. Đánh giá kết quả bài thực hành

TÀI LIỆU THAM KHẢO

[1] Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*.