BỘ THÔNG TIN VÀ TRUYỀN THÔNG

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

------o0o------

BÁO CÁO BÀI THỰC HÀNH

PHÂN TÍCH FILE ELF

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

DANH MỤC CÁC HÌNH VẼ

DANH MỤC CÁC BẢNG BIỂU

DANH MỤC VIẾT TẮT

Từ viết tắt	Thuật ngữ tiếng Anh /Giải thích	Thuật ngữ tiếng Việt/Giải thích
ELF	Executable and Linkable Format	Định dạng file thực thi và liên kết trong hệ điều hành Unix/Linux.
lvl	Level	Cấp độ hoặc mức độ trong bài tập hoặc thử thách.
	Section	Phần trong file ELF chứa dữ liệu hoặc mã lệnh, ví dụ: .text, .data, .bss.
	File	Tập tin được sử dụng trong bài phân tích, ví dụ: file ELF hoặc DLL.
	Header	Phần đầu của file ELF chứa thông tin về định dạng, kiến trúc và cấu trúc file.

1.1. Giới thiệu bài thực hành

Bài thực hành " **Phân tích file ELF** " được thiết kế nhằm giúp sinh viên hiểu rõ hơn về cấu trúc file ELF (Executable and Linkable Format) và các kỹ thuật phân tích file ELF cơ bản. Đây là một phần quan trọng trong lĩnh vực phân tích mã độc, đặc biệt khi xử lý các chương trình được biên dịch cho hệ điều hành Linux hoặc các hệ thống nhúng. Việc nắm vững cấu trúc và cách thức hoạt động của file ELF sẽ là bước khởi đầu cần thiết để sinh viên làm quen với các quy trình phân tích và nhận thức rõ về các đặc điểm kỹ thuật của file thực thi.

Trong bài thực hành này, sinh viên sẽ phân tích file ELF dựa trên các bài tập thực tiễn từ **Level 2** và **Level 3** của tài liệu "Practical Binary Analysis". Ở Level 2, sinh viên sẽ học cách sử dụng công cụ để kiểm tra các section trong file ELF và tìm các chuỗi ký tự hoặc thông tin mã hóa nằm trong vùng .rodata. Qua đó, sinh viên sẽ nắm được cách phân tích và giải mã thông tin mà không cần thực thi file, đảm bảo tính an toàn cho môi trường phân tích.

Trong Level 3, sinh viên sẽ tiến hành chỉnh sửa header ELF để khắc phục lỗi kiến trúc không tương thích (architecture mismatch), qua đó làm cho file ELF có thể được phân tích hoặc chạy trong môi trường hiện tại. Kỹ thuật này giúp sinh viên hiểu rõ hơn về cách file ELF tổ chức thông tin và cách xử lý các tình huống lỗi thường gặp khi làm việc với file thực thi.

Việc phân tích file ELF không chỉ giúp sinh viên nhận diện được các đặc điểm bất thường mà còn trang bị cho họ kỹ năng cần thiết để xử lý các file thực thi phức tạp. Bài thực hành này đồng thời giúp sinh viên làm quen với các công cụ như readelf, objdump, bvi và các phương pháp chỉnh sửa header file, nhằm phục vụ tốt hơn cho việc điều tra mã độc và nâng cao an toàn hệ thống.

Thông qua bài thực hành, sinh viên sẽ hiểu cách phân tích cấu trúc file ELF một cách hiệu quả, nhận thức được tầm quan trọng của việc áp dụng các kỹ thuật phân tích tiên tiến, và nâng cao kỹ năng xử lý các tình huống thực tế trong lĩnh vực bảo mật thông tin. Đây là một bước chuẩn bị quan trọng cho các công việc liên quan đến phân tích tĩnh 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 file ELF và cách sử dụng các công cụ phân tích tĩnh như objdump, bvi và readellf. Sinh viên sẽ học cách phân tích cấu trúc của file ELF, bao gồm các section, header, và các thông tin bên trong vùng .rodata. Qua bài thực hành này, sinh viên sẽ hiểu cách tìm kiếm chuỗi ký tự mã hóa, chỉnh sửa header ELF để khắc phục các lỗi về kiến trúc không tương thích, và phát hiện các đặc điểm đáng ngờ trong file ELF. Mục tiêu là giúp sinh viên phát triển kỹ năng phân tích tĩnh một cách có hệ thống và an toàn.

1.2.2. Yêu cầu đối với sinh viên

- Hệ điều hành Linux.
- Cấu trúc file ELF và cách thức hoạt động của các section như .rodata, .text, và .data.
- Các công cụ chỉnh sửa header để thay đổi thông tin về kiến trúc.

1.2.3. Nội dung thực hành

Khởi động bài lab:

Vào terminal, gõ:

labtainer -r mal-hder-elf

(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-hder-elf. Trên terminal ***mal-hder-elf*** đã được cài đặt sẵn công cụ để thực hiện nhiệm vụ cũng như được cài đặt sẵn một file oracle để làm và giải thu thập lấy file lvl2, lvl3 để hoàn thành bài lab.

Sau khi terminal hiện lên, ta sẽ thấy 1 vài file như là lvl2 và oracle. Sau đó chạy file oracle với đoạn mã đã cho để tải xuống file lvl2. Sau đó thực hiện các nhiệm vụ của bài thực hành.

- Sinh viên cấp quyền và chạy file lvl2 để xem kết quả tạo ra. Thấy được mỗi kết quả của mỗi lần chạy các số thập lục phân khác nhau nên từ đó có thể

phán đoán được các chuỗi ký tự có thể tìm thấy được ở trong data của file lvl2.

objdump -s --section .rodata ./lvl2

- Sinh viên sử dụng công cụ objdump để đọc các section data của file là thu về đoạn ký tự để mở khóa lvl3.

./oracle 034fc4f6a536f2bf74f8d6d3816cdf88

- Sinh viên thực hiện chạy file tương tự như lvl2, nhưng kết quả khi chạy file lvl3 thì lại là không thể chạy được. Sau đó dùng lệnh file để kiểm tra xem định dạng và hệ thống của nó là gì.
- Thấy được rằng nó là 64-bit nhưng hệ thống máy ảo là x86-64 nên ta phải sửa cho nó chạy được như 1 lệnh elf bình thường.

file lvl3

- Sinh viên sử dụng lệnh readelf để xem tiêu đề của file elf. Và từ đây thấy được rằng tiêu đề của nó bị sửa phá hỏng có vẻ là cố ý. Vì thấy được rằng mục **Start of program headers** chứa một giá trị quá lớn.

readelf -h lvl3

- Dùng công cụ BVI để đọc header file elf và chỉnh sửa đoạn sai. Để sửa được đoạn này thì sinh viên cần hiểu về header của 1 file elf trước.

bvi lvl3

- Sinh viên tìm tới đoạn mã bị sai và sửa lại chúng. e_ident[EI_OSABI] từ 0x0b thành 0x00, e_machine từ 0x34 thành 0x3e và e_phoff từ 0xdeadbeef thành 0x40
- Sau đó lưu lại và thoát ra ngoài. Kiểm tra lại xem header của file đã về đúng định dạng chúng ta muốn là 64 thay vì một chuỗi ký tự quá lớn hay chưa. Sau đó chạy file để lấy kết quả lần nữa.
- Kết quả vẫn chưa chuẩn nên sinh viên cần đọc dữ liệu ở .rodata như lvl2 và thấy được cụm md5sum, thử md5sum flag được hiện ra khi chạy file nhưng kết quả thu được vẫn là flag lỗi.

- Sinh viên dùng **ltrace** để kiểm tra lại file lvl3 xem nó hoạt động như thế nào và thấy được câu lệnh **md5sum ./lvl3**, thử chạy câu lệnh nhưng vẫn thu được flag sai.

ltrace -f ./lvl3

- Từ đó có thể đoán được điều cần làm tiếp để có thể thu được flag là phải chỉnh sửa ở tệp elf một lần nữa. Kiểm tra các section của file một lần nữa, khi chú ý vào thì thấy được rằng nó bị thiếu phần **.text.**

objdump -d -Mintel lvl3|grep section

- Dùng objdup cũng không thể tìm được phần còn thiếu. Sinh viên dùng sang công cụ readelf và ta thấy được nó bị sai ở một trường giá trị là **NOBITS** trong khi đúng ra nó phải là **PROGBITS,** vì nó chứa mã thực thi.

readelf -S lvl3|grep text

- Và để sửa được nó về giá trị đúng thì sinh viên cần tính toán được vị trí cần sửa của nó. Và để làm được thì đầu tiên sinh viên cần xác định thứ tự của .text section.

readelf -S --wide lvl3

- Sau khi tính toán được vị trí thì sinh viên tiến hành sửa lại sao cho về lại **PROGBITS** sau cho chạy lại lệnh readelf để kiểm tra lần cuối.

bvi -s 0x1500 ./lvl3

readelf -S --wide lvl3|grep .text

- Kết hợp tất cả với thông tin về câu lệnh mà ta thu thập được ở ệnh **ltrace** thì chạy lệnh để lấy được flag và chạy file oracle với giá trị đó để kết thúc bài lvl3 và mở khoá lvl4.

md5sum ./lvl3

./oracle 3a5c381e40d2fffd95ba4452a0fb4a40

- 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-hder-elf

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-hder-elf

1.3. Phân tích yêu cầu bài thực hành

Bài thực hành yêu cầu sinh viên làm quen với việc phân tích file ELF, một định dạng file thực thi phổ biến trên hệ điều hành Linux. Sinh viên sẽ sử dụng các công cụ như readelf, objdump, bvi để khám phá cấu trúc file ELF, thu thập thông tin về các section, chuỗi ký tự và các đặc điểm quan trọng khcas. Mục tiêu của bài thực hành là giúp sinh viên hiểu rõ cách phân tích file ELF một cách an toàn mà không cần thực thi file, qua đó nhận diện và đánh giá các vấn đề tiềm ẩn trong cấu trúc của file.

Sinh viên sẽ bắt đầu bài thực hành bằng lệnh khởi tạo labtainer <tên bài lab> và nhập MSV của mình. Sau đó, sinh viên sẽ sử dụng các công cụ như readelf, objdump, bvi đẻ phân tích nội dung các section đặc biệt là phân cùng .rodata. Trong lvl2, sinh viên sẽ tìm kiếm các chuỗi ký tự ẩn trong vùng dữ liệu và sẽ thực hiện giải mã để tìm ra các thông tin quan trọng. Trong lvl3 thì sinh viên sẽ chỉnh sửa header ELF để khắc phục lỗi không tương thích kiến trúc, đảm bảo file có thể phân tích hoặc chạy được trong môi trường hiện tại.

Cuối cùng sau khi hoàn thành các bước thì file oracle sẽ tạo ra một file lvl4. Kết thúc bài thực hành này giúp sinh viên phát triển kỹ năng phân tích file ELF, từ đó nâng cao khả năng xử lý file thực thi và bảo mật thông tin trong thực tế.

1.4. Thiết kế bài thực hành

Trên môi trường máy ảo Ubuntu được cung cấp, sử dụng docker tạo ra 1 container mang tên mal-hder-elf.

Cấu hình docker gồm có:

- mal-hder-elf: Lưu cấu hình cho máy
- config: Lưu cấu hình hoạt động của hệ thống
- dockerfile: Mô tả cấu hình của 1 container trong đó:
    - mal-hder-elf: sử dụng các thư viện mặc định của hệ thống. Tích hợp sẵn mẫu file thực thi cho sẵn để thực hiện các nhiệm vụ.

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

- Phân tích vùng .rodata của lvl2 để lấy ký tự mã hoá.
- Chỉnh sửa header ELF để khắc phục lỗi kiến trúc không tương thích của lvl3.
- Sử dụng ltrace để đọc mã file lvl3.
- Thu thập và chỉnh sửa thông tin của .text.
- Kiểm tra lại file ELF và chạy theo câu lệnh chuẩn.

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

lvl3-start	mal-hder-elf	.bash_history	CONTAINS	./oracle 034fc4f6a536f2bf74f8d6d3816cdf88	File
lvl3-done	mal-hder-elf	.bash_history	CONTAINS	./oracle 3a5c381e40d2fffd95ba4452a0fb4a40	File
rdata-lvl2	mal-hder-elf	.bash_history	CONTAINS	objdump -s --section .rodata ./lvl2	File
r-elf	mal-hder-elf	.bash_history	CONTAINS	readelf -h lvl3	File
fix-hxd	mal-hder-elf	readelf.stdout	CONTAINS	Start of program headers: 64 (bytes into file)	File
text-hid	mal-hder-elf	.bash_history	CONTAINS	readelf -S lvl3	File
fix-text	mal-hder-elf	readelf.stdout	CONTAINS	[14] .text PROGBITS 0000000000400550 000550 0001f2 00 AX 0 0 16	File

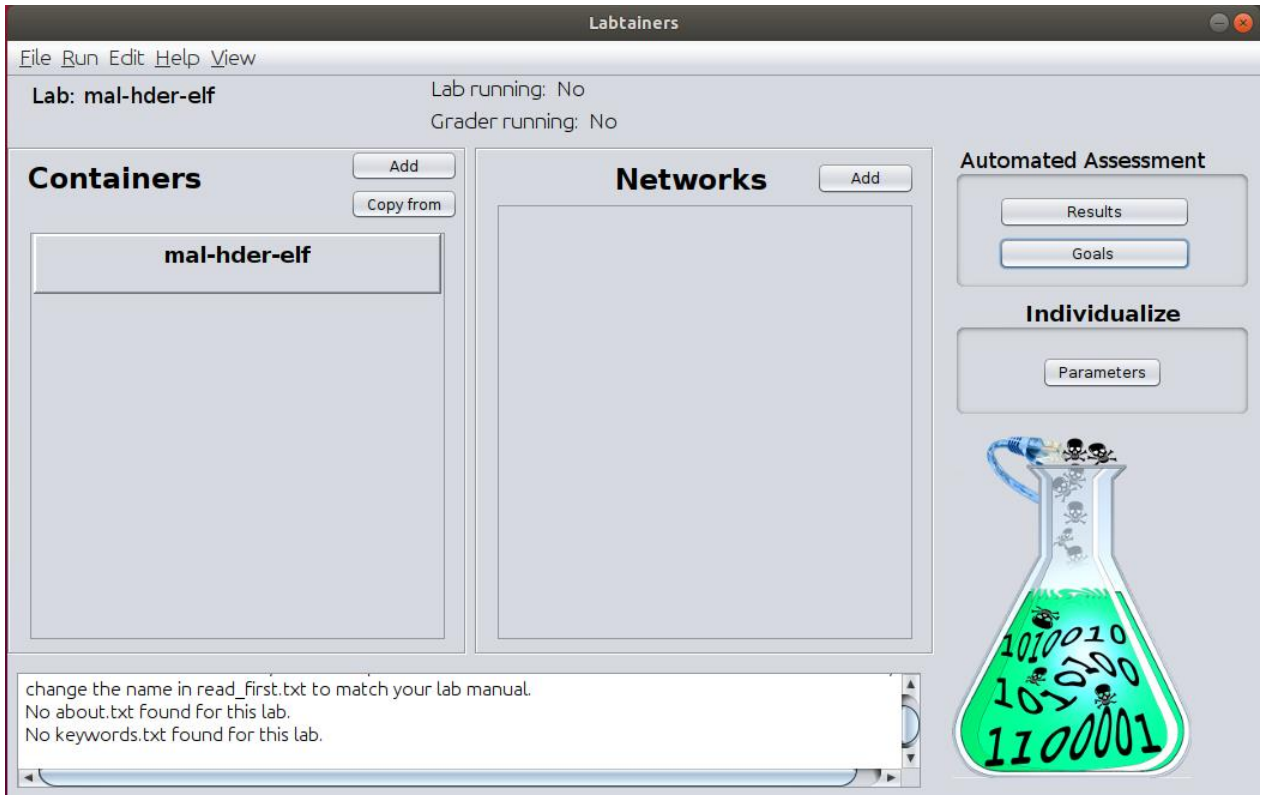
Bảng 2. Bảng Goals

Goal ID	Goal Type	Boolean	Boolean Result Tags

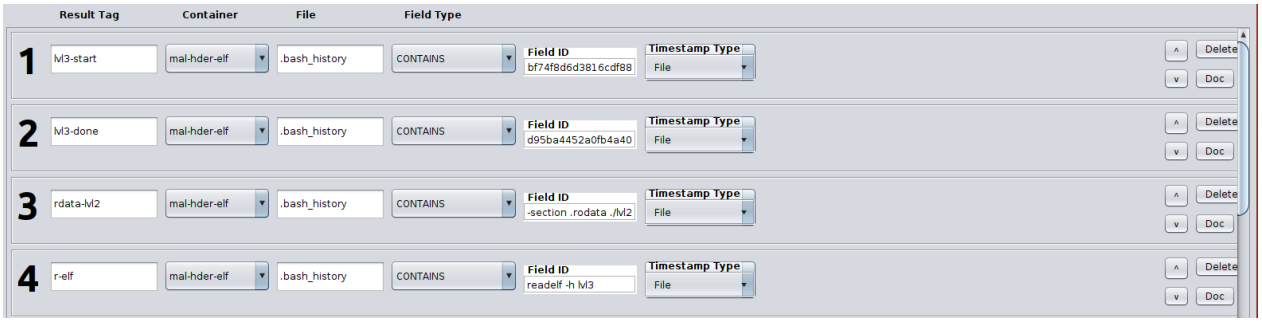
Bảng 3. Bảng Paramater

Param ID	Operator	File name	Symbol	Step	Hashstring

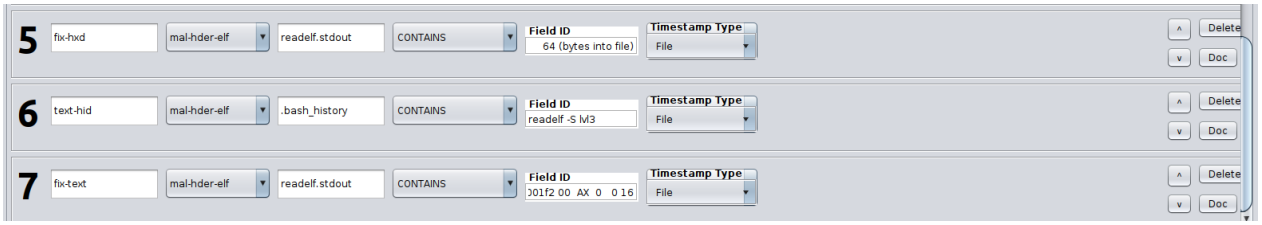
1.5. Cài đặt và cấu hình các máy ảo



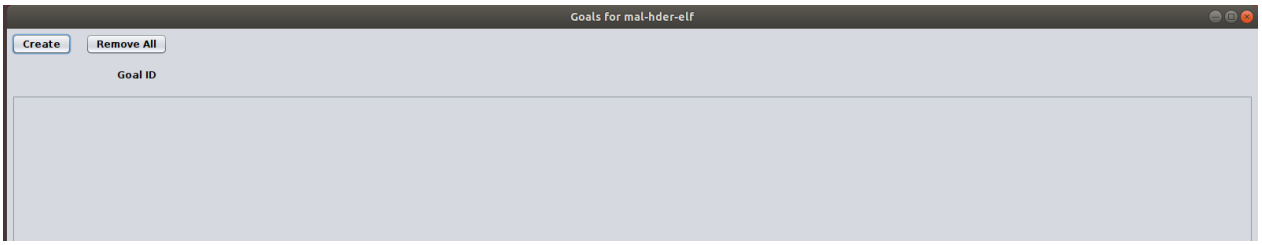
Hình 1: Giao diện Labedit



Hình 2: Cài đặt Result 1



Hình 3: Cài đặt phần Result 2



Hình 4: Cài đặt phần Goals

Hình 5: Cài đặt phần Parameter

```
#
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
#
# Install files in the user home directory
#
ADD $labdir/$imagedir/home_tar/home.tar $HOME
ADD $labdir/$imagedir/sys_tar/levels.db $HOME
ADD $labdir/$imagedir/sys_tar/oracle $HOME
ADD $labdir/$imagedir/sys_tar/libssl1.0.0_1.0.2n-1ubuntu5.3_amd64.deb $HOME
ADD $labdir/$imagedir/sys_tar/payload $HOME
RUN sudo dpkg -i $HOME/libssl1.0.0_1.0.2n-1ubuntu5.3_amd64.deb
RUN rm -f $HOME/libssl1.0.0_1.0.2n-1ubuntu5.3_amd64.deb
# remove after docker fixes problem with empty tars
RUN rm -f $HOME/home.tar
ADD $labdir/$lab.tar.gz $HOME
#
#  The first thing that executes on the container.
#
USER root
CMD ["/bin/bash", "-c", "exec /sbin/init --log-target=journal 3>&1"]
```

Hình 6: Dockerfiles

1.6. Tích hợp và triển khai

1.6.1. Docker Hub

```
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 minhh310
student@ubuntu:~/labtainer/trunk/labs$ git config --global user.email hieupvm310@gmail.com
student@ubuntu:~/labtainer/trunk/labs$ git add mal-hder-elf
student@ubuntu:~/labtainer/trunk/labs$ git commit mal-hder-elf -m "Adding an IModule"
[master e824bf2] Adding an IModule
 23 files changed, 729 insertions(+)
 create mode 100755 mal-hder-elf/config/mal-hder-elf-home_tar.list
 create mode 100755 mal-hder-elf/config/parameter.config
```

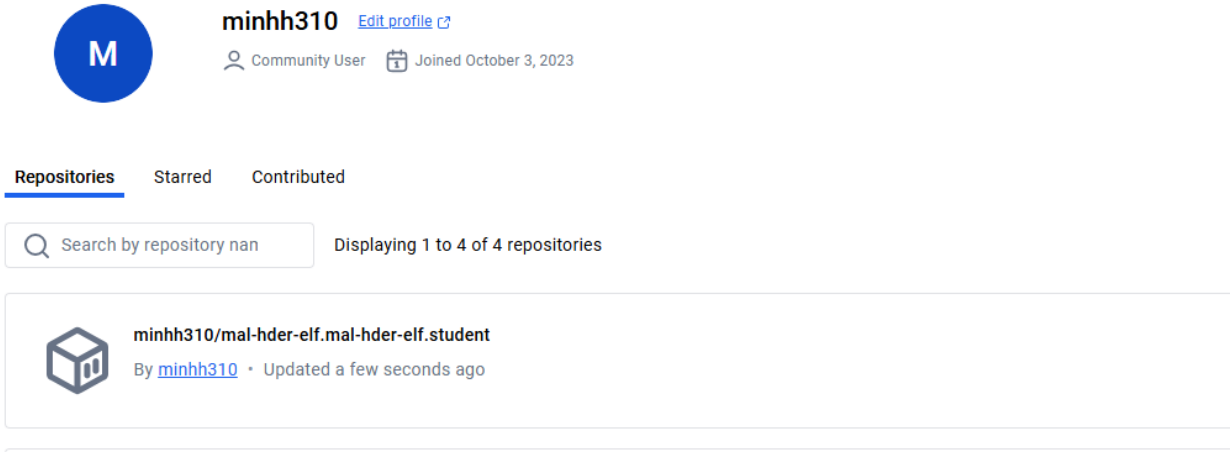
Hình 7: Add và comit bài lab

```
student@ubuntu:~/labtainer/trunk/labs$ git commit mal-hder-elf -m "Adding an IModule"
[master e824bf2] Adding an IModule
 23 files changed, 729 insertions(+)
 create mode 100755 mal-hder-elf/config/mal-hder-elf-home_tar.list
 create mode 100755 mal-hder-elf/config/parameter.config
 create mode 100755 mal-hder-elf/config/start.config
 create mode 100755 mal-hder-elf/config/stealthfile-home_tar.list
 create mode 100755 mal-hder-elf/dockerfiles/.Dockerfile.stealthfile.stealthfile.student.swp
 create mode 100755 mal-hder-elf/dockerfiles/Dockerfile.mal-hder-elf.mal-hder-elf.student
 create mode 100755 mal-hder-elf/docs/read_first.txt
 create mode 100755 mal-hder-elf/instr_config/goals.config
 create mode 100755 mal-hder-elf/instr_config/pregrade.sh
 create mode 100755 mal-hder-elf/instr_config/results.config
 create mode 100755 mal-hder-elf/mal-hder-elf/_bin/fixlocal.sh
 create mode 100755 mal-hder-elf/mal-hder-elf/_bin/treataslocal
 create mode 100755 mal-hder-elf/mal-hder-elf/_system/etc/login.defs
 create mode 100755 mal-hder-elf/mal-hder-elf/_system/etc/securetty
 create mode 100755 mal-hder-elf/mal-hder-elf/home_tar/home.tar
 create mode 100644 mal-hder-elf/mal-hder-elf/mal-hder-elf.mal-hder-elf.student.tar.gz
 create mode 100644 mal-hder-elf/mal-hder-elf/sys_mal-hder-elf.mal-hder-elf.student.tar.gz
 create mode 100755 mal-hder-elf/mal-hder-elf/sys_tar/.check_res.sh
 create mode 100755 mal-hder-elf/mal-hder-elf/sys_tar/levels.db
 create mode 100755 mal-hder-elf/mal-hder-elf/sys_tar/libssl1.0.0_1.0.2n-1ubuntu5.3_amd64.deb
 create mode 100755 mal-hder-elf/mal-hder-elf/sys_tar/oracle
 create mode 100755 mal-hder-elf/mal-hder-elf/sys_tar/payload
 create mode 100755 mal-hder-elf/mal-hder-elf/sys_tar/sys.tar
student@ubuntu:~/labtainer/trunk/labs$
```

Hình 8: Thêm bài lab muốn lưu trữ

```
student@ubuntu:~/labtainer/trunk/distrib$ ./publish.py -d -l mal-hder-elf
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]
adding [shellbasics]
adding [ldaptst]
adding [mariadbtst]
```

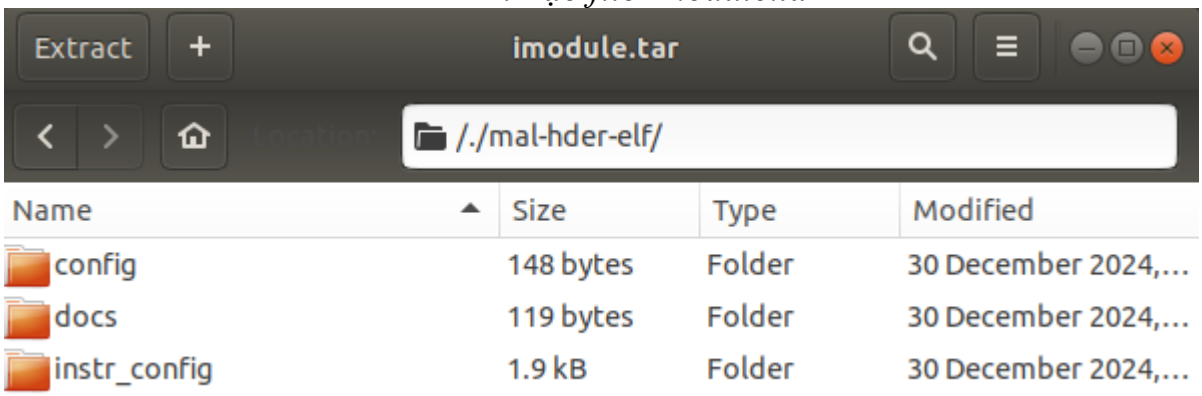
Hình 9: Quá trình tải bài lab lên dockerhub



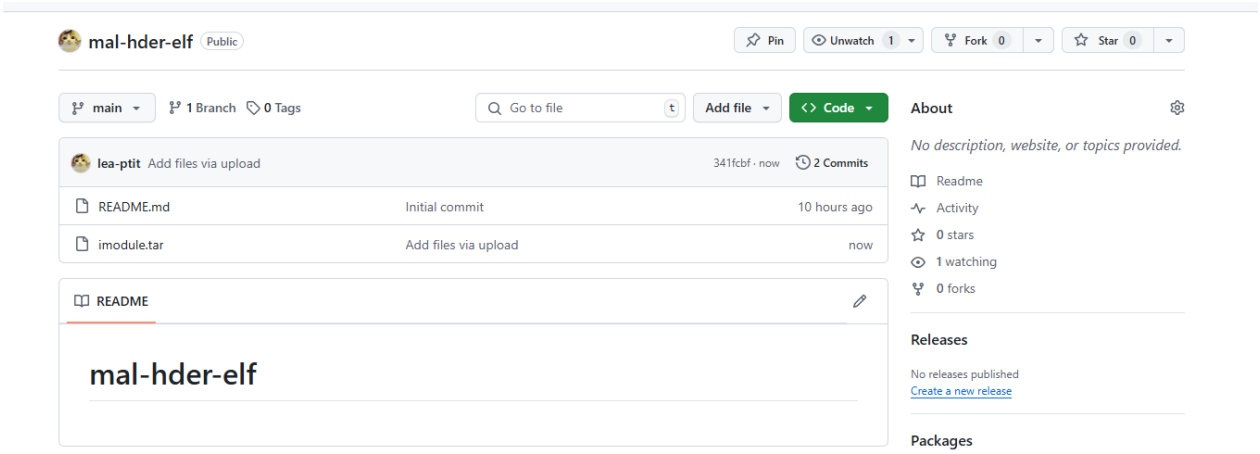
Hình 10: Đẩy thành công

1.6.2. Github

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

Trước khi bắt đầu thì sinh viên tải labtainer từ trên github về với câu lệnh

imodule https://github.com/lea-ptit/mal-hder-elf/raw/main/imodule.tar

```
student@ubuntu:~/labtainer/labtainer-student$ imodule https://github.com/lea-ptit/mal-hder-elf/raw/main/imodule.tar
Adding imodule path https://github.com/lea-ptit/mal-hder-elf/raw/main/imodule.tar
Updating IModule from https://github.com/lea-ptit/mal-hder-elf/raw/main/imodule.tar
```

Hình 14: Tải bài thực hành từ github

Sau đó chạy lệnh labtainer -r mal-hder-elf để khởi động lab và làm bài thực hành.

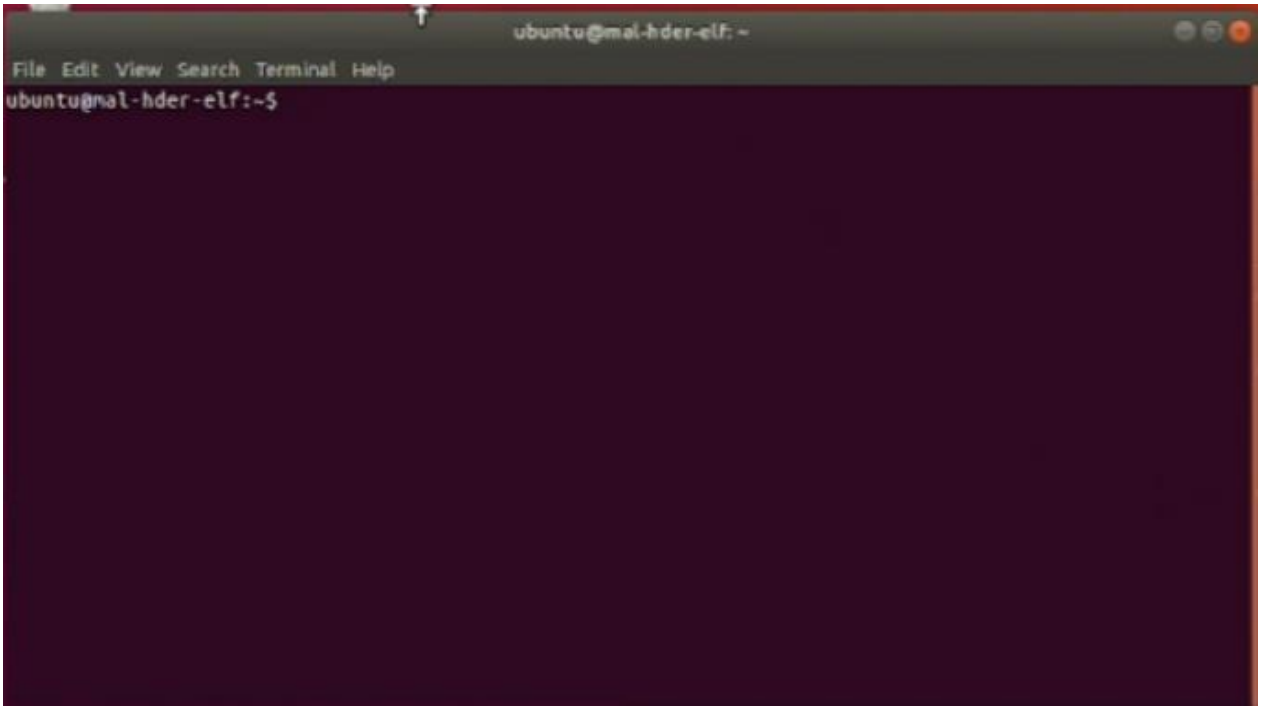
```
student@ubuntu:~/labtainer/labtainer-student$ labtainer -r mal-hder-elf
latest: Pulling from minhh310/mal-hder-elf.mal-hder-elf.student
d096daa29c9f: Pulling fs layer
466b763df16e: Pulling fs layer
d096daa29c9f: Pull complete
466b763df16e: Pull complete
bfa4df29d204: Pull complete
b0bd15ce77e4: Pull complete
43b1da0bca37: Pull complete
5ab54175e338: Pull complete
5c54d6003fa1: Pull complete
90dbb2dc809a: Pull complete
58573acd3106: Pull complete
59edfba1a5d8: Pull complete
1302dae80c8b: Pull complete
71b90a4cae6d: Pull complete
e8c82e289b50: Pull complete
3fa77e25a60e: Pull complete
e58050f37bdd: Pull complete
a3362d1c42e3: Pull complete
3c7b88116cf9: Pull complete
6b5a151afeb1: Pull complete
Digest: sha256:13e60a74d72c3ea19822d1d7e4e797498c7c1b25b73cec6dd475df8cb6e124d7
Status: Downloaded newer image for minhh310/mal-hder-elf.mal-hder-elf.student:latest

Please enter your e-mail address:  [B20DCAT061]
Started 1 containers, 0 completed initialization, please wait...
```

Hình 15: Khởi động bài thực hành sau khi tải lab về

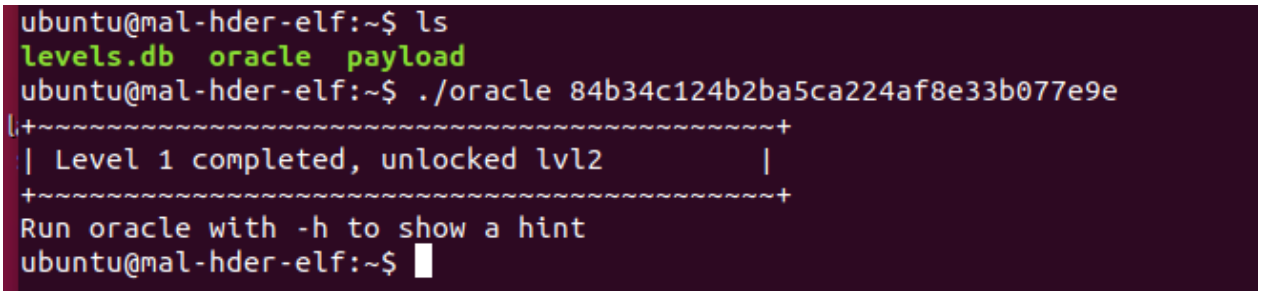
Bài thực hành đã được xây dựng thành công, dưới đây là hình ảnh minh họa về bài thực hành:

Khi bắt đầu labtainer mal-hder-elf, ta sẽ thấy có 1 terminal nhảy lên.



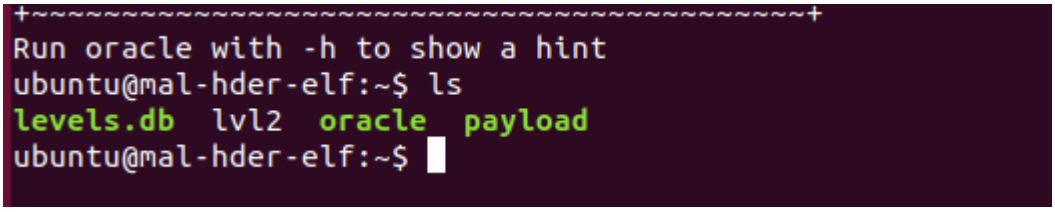
Hình 16: Terminal của bài thực hành

Sau khi terminal hiện lên thì ta chạy file ./oracle với đoạn mã cho sẵn để thu được file lvl2 để bắt đầu nhiệm vụ. *./oracle 84b34c124b2ba5ca224af8e33b077e9e*



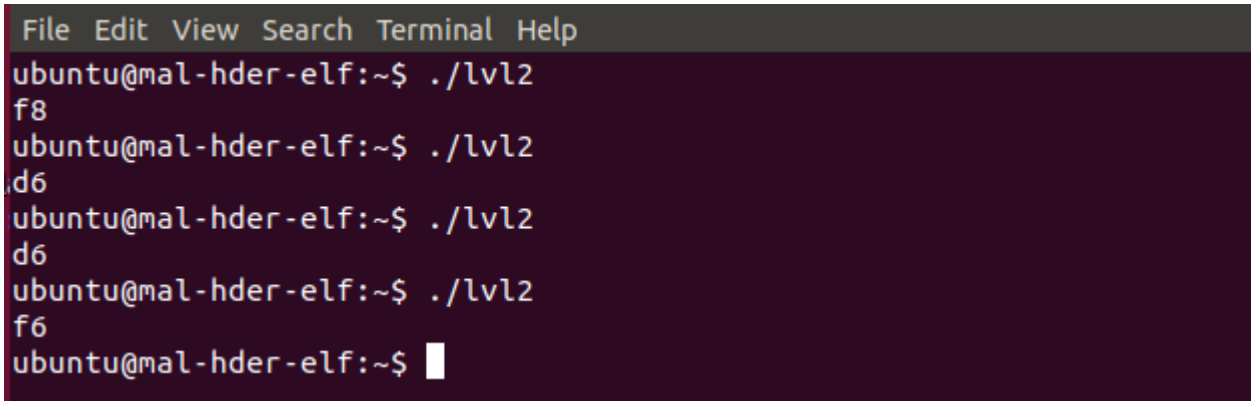
Hình 17: Mở khoá lvl2

Sau khi chạy xong thì ta thu được bài thực hành với những file như bên dưới.



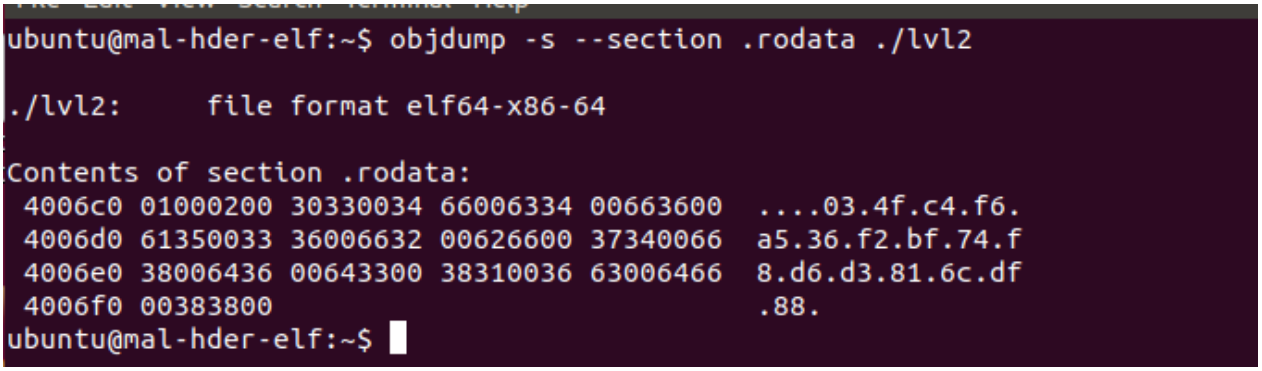
Hình 18: Thành phần của bài thực hành

Sau cấp quyền và chạy file lvl2 thì ta thấy được mỗi một lần chạy file thực thi thì sẽ thấy được 1 kết quả khác nhau dưới dạng thập lục phân.



Hình 19: Kết quả khi chạy lvl2

Từ đó phán đoán được thông tin cần tìm được lưu ở data của file thực thi lvl2. Đọc data của file này ta thu được kết quả cần thiết để mở khoá lvl3.



Hình 20: .rodata của lvl2

Sau đó chạy file ./oracle với chuỗi ký tự thu được thì ta mở khoá được lvl3.

./oracle 034fc4f6a536f2bf74f8d6d3816cdf88

```
ubuntu@mal-hder-elf:~$ ./oracle 034fc4f6a536f2bf74f8d6d3816cdf88
+-----------------------------------------------+
| Level 2 completed, unlocked lvl3              |
+-----------------------------------------------+
Run oracle with -h to show a hint
ubuntu@mal-hder-elf:~$
```

Hình 21: Mở khoá lvl3

Sau khi cấp quyền thì không chạy được như lvl2.

```
ubuntu@mal-hder-elf:~$ ls
levels.db  lvl2  lvl3  oracle  payload
ubuntu@mal-hder-elf:~$ chmod +x lvl3
ubuntu@mal-hder-elf:~$ ./lvl3
/usr/sbin/exec_wrap.sh: line 16: ./lvl3: cannot execute binary file: Exec format error
ubuntu@mal-hder-elf:~$
```

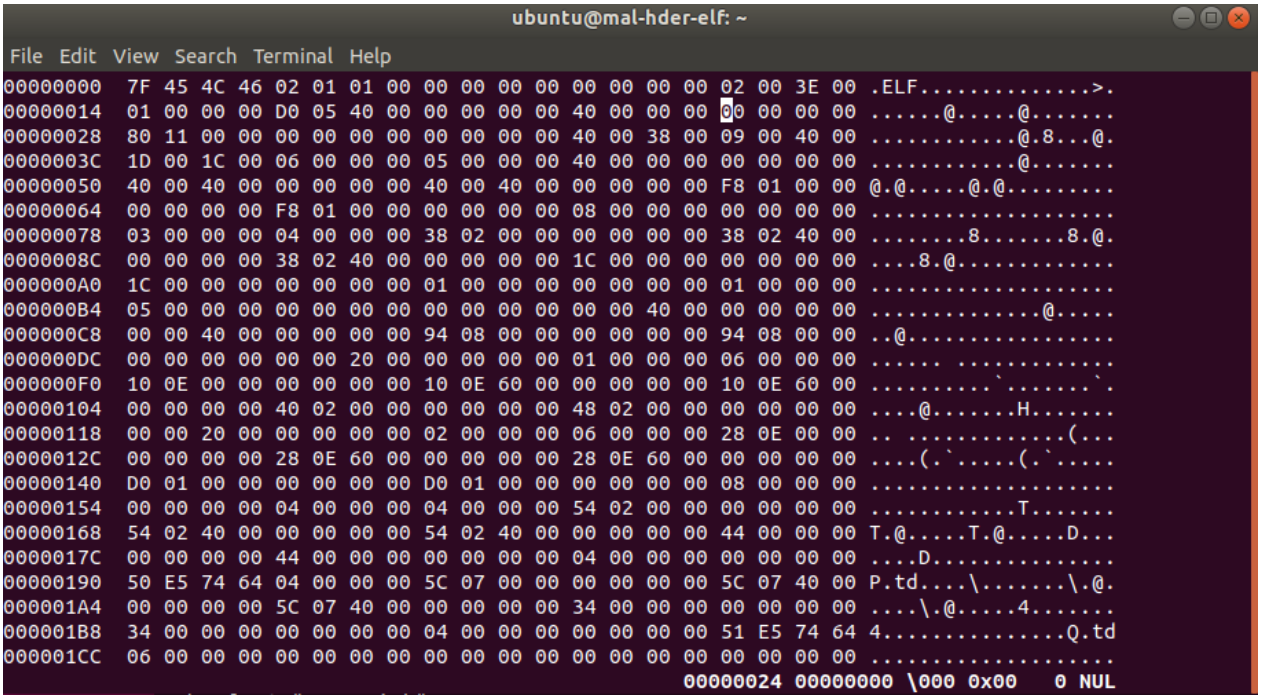
Hình 22: Cấp quyền và run file lvl3

Xác định lại định dạng file và sửa lại cho đúng với môi trường đang thực hành để chạy được file.

```
ubuntu@mal-hder-elf:~$ readelf -h lvl3
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 0b 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            Novell - Modesto
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Motorola Coldfire
  Version:                           0x1
  Entry point address:               0x4005d0
  Start of program headers:          4022250974 (bytes into file)
  Start of section headers:          4480 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         9
  Size of section headers:           64 (bytes)
  Number of section headers:         29
  Section header string table index: 28
readelf: Error: Reading 504 bytes extends past end of file for program headers
ubuntu@mal-hder-elf:~$
```

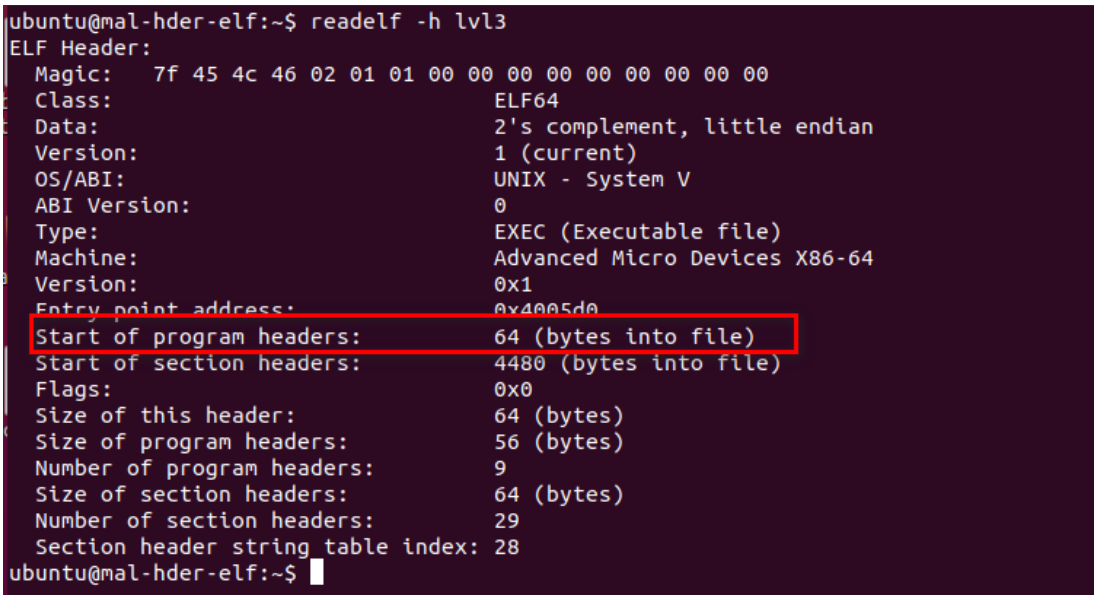
Hình 23: Header của lvl3

Ta thấy được header của tệp này đang bị hỏng, vì mục Start of program headers chắc chắn không thể lớn như vậy. Dùng bvi để sửa lại header của file thực thi elf.



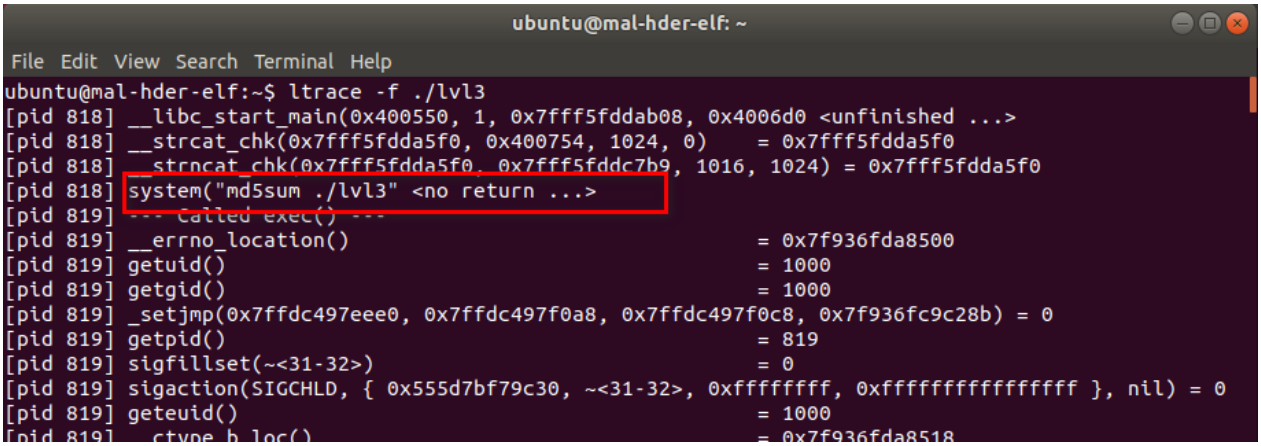
Hình 24: Sửa header elf

Sau khi sửa xong chạy lại lệnh readelf thì ta thấy được nó đã về là 64. Lúc này đã có thể chạy file trong môi trường hiện tại.



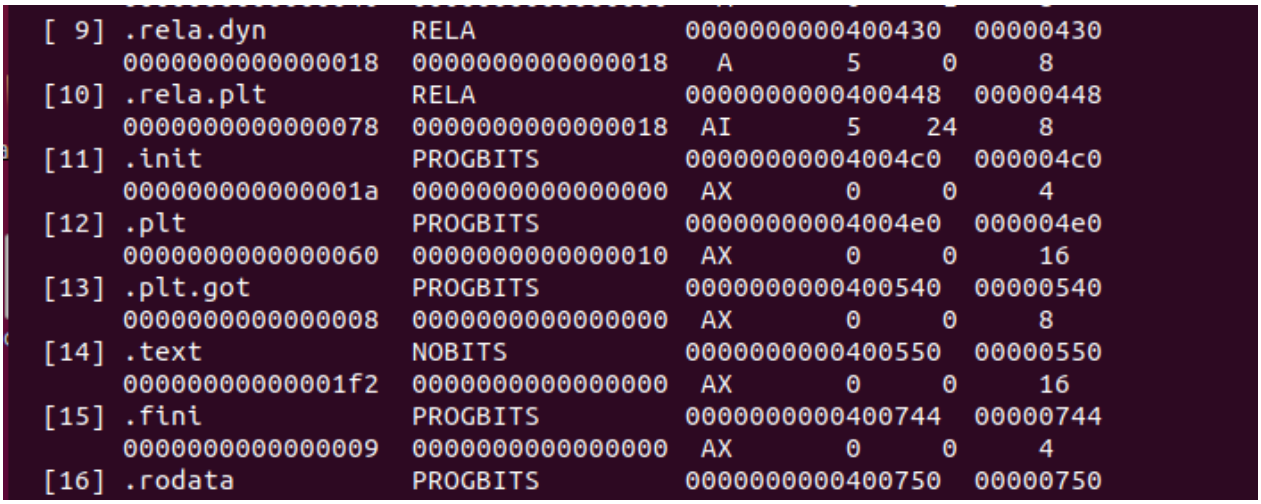
Hình 25: Header file sau khi sửa

Ta chạy file lvl3 và thu được mỗi chuỗi ký tự. Nhưng khi dùng chuỗi đấy cho file ./oracle thì ta thấy được là nó bị sai. Dùng công cụ **ltrace** để xem có chuyện gì xảy ra mà nó lại không ra được chuỗi đúng. Khi dùng ltrace thì ta thấy được rằng file thực thi phải chạy theo cú pháp **md5sum ./lvl3.**



Hình 26: ltrace của lvl3

Nhưng dù chạy đungs theo cách mà chương trình đưa ra nhưng vẫn không thể mở khoá lvl4 với chuỗi ký tự đấy. Từ đấy có thể đoán được rằng chương trình được sửa vẫn chưa hoàn thiện, vẫn còn phải sửa tiếp. Và khi kiểm tra các section thì ta thấy sự vắng mặt của phần .text qua câu lệnh *objdump -d -Mintel lvl3|grep section.* Và khi ta liệt kê hết ra qua lệnh *readelf -S lvl3* thì ta thấy được rằng section .text đang là trường **NOBITS** thay vì là **PROGBITS** vì nó chứa mã thực thi.

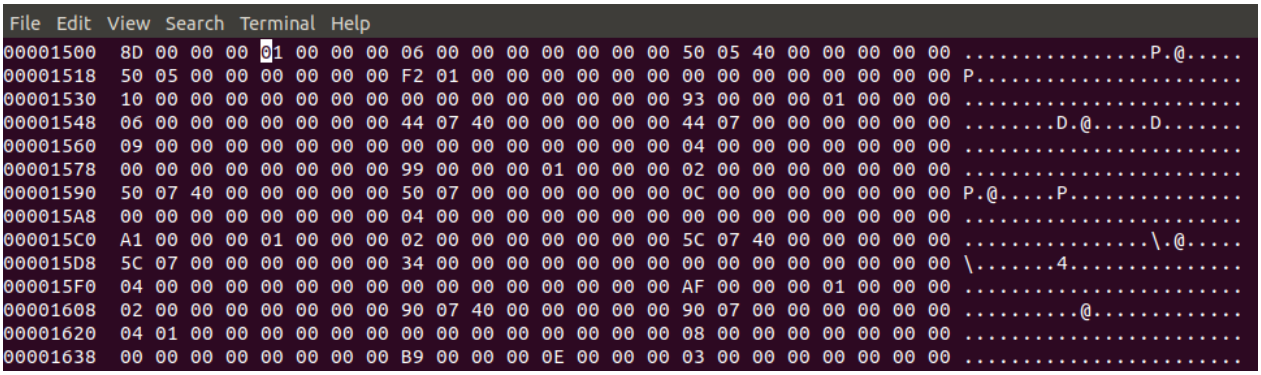


Hình 27: Section .text của lvl3

Ta tiến hành tính toán vị trí của .text để dùng bvi sửa đổi cho về lại dạng chuẩn.

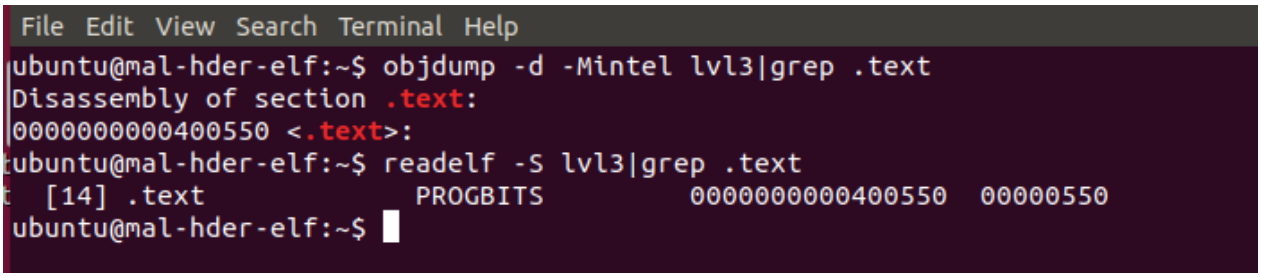
Ta thấy có 29 section và bắt đầu từ offet 0x1180. Và phần text là mục thứ 14 trong phần tất cả tiêu đề nên có giá trị hex là 0xe. Tiêu dề dài 64 byte (0x40), từ đó tính được độ lệch của .text sẽ bắt đầu từ 0x1180 + 0xe * 0x40 = 0x1500.

Dùng bvi để chỉnh sửa **bvi -s 0x1500 ./lvl3**. Và chỉnh từ 0x8 về lại 0x1.



Hình 28: Chỉnh sửa .text

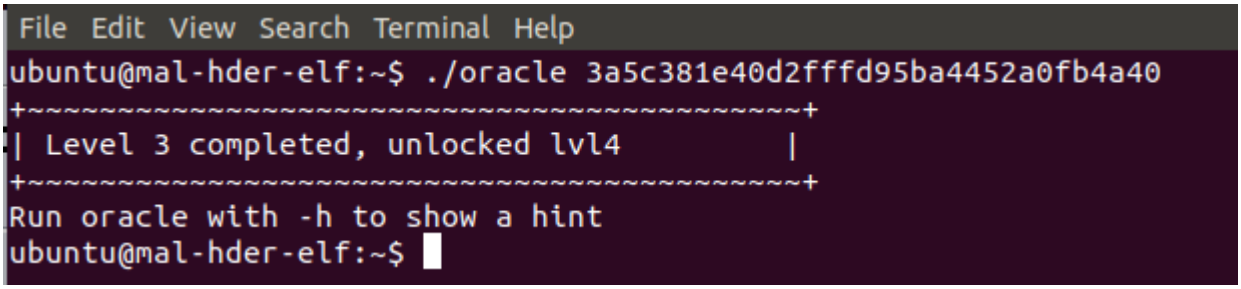
Ta dùng lệnh *objdump -d -Mintel lvl3|grep .text* và *readelf -S --wide lvl3|grep .text* để check lại lần cuối xem phân vùng .text đã là **PROGBITS** hay chưa.



Hình 29: Kiểm tra .text

Sau khi kiểm tra thấy tất cả đã đúng thì ta chạy lại lệnh **md5sum** một lần nữa và thu về chuỗi giá trị để mở khoá lvl4, kết thúc lvl3.

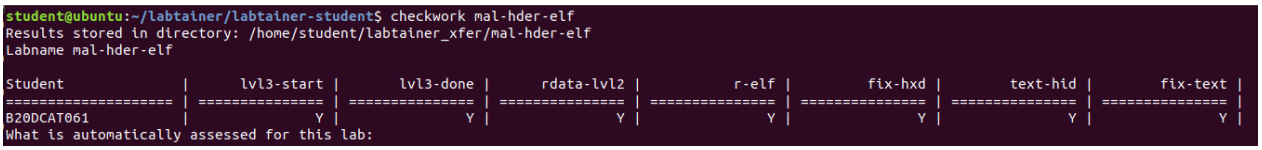
./oracle 3a5c381e40d2fffd95ba4452a0fb4a40



Hình 30: Giải mã thành công lvl4

Sau khi làm xong hết tất cả thì ta tiến hành checkwork.

Tiến hành checkwork



Hình 31: Checklwork của bài thực hành

TÀI LIỆU THAM KHẢO

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