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

BÁO CÁO BÀI THỰC HÀNH
HỌC PHẦN: KỸ THUẬT GIẤU TIN
MÃ HỌC PHẦN: INT14102

NHÓM LỚP: D21CQAT03-B

TÊN BÀI: TÁCH TIN TRONG TỆP ÂM THANH BẰNG KỸ THUẬT FHSS

Sinh viên thực hiện:

B21DCAT135 Đặng Quý Nam

Giảng viên: PGS.TS. Đỗ Xuân Chợ

HỌC KỲ 2 NĂM HỌC 2024-2025

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ 3
DANH MỤC CÁC BẢNG BIỂU 3
CHƯƠNG 1: GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH 4
1.1 Giới thiệu chung về bài thực hành 4
1.2 Nội dung và hướng dẫn bài thực hành 4
CHƯƠNG 2: PHÂN TÍCH YÊU CẦU BÀI THỰC HÀNH 7
2.1 Thiết kế bài thực hành 7
2.2 Cài đặt và cấu hình máy ảo 7
2.3 Tích hợp và triển khai 8
CHƯƠNG 3: THỬ NGHIỆM VÀ ĐÁNH GIÁ. 11
TÀI LIỆU THAM KHẢO 17

DANH MỤC CÁC HÌNH VẼ

Hình 1 File result.config....7
Hình 2 Giao diện labedit....8
Hình 3 Giao diện result.config....8
Hình 4 Đẩy image bài thực hành lên docker hub....9
Hình 5 Image trên dockerhub....9
Hình 6 Tạo file imodule....10
Hình 7 File imodule trên github....10
Hình 8 Cài đặt bài thực hành....11
Hình 9 Khởi động badi lab....12
Hình 10 Địa chỉ IP máy sender....12
Hình 11 Địa chỉ IP máy receiver....13
Hình 12 Cấu hình SSH máy receiver....13
Hình 13 Khởi động lại SSH....14
Hình 14 Gửi file âm thanh sang receiver....14
Hình 15 Gửi file checksum sang receiver....14
Hình 16 Tách tin từ file âm thanh....14
Hình 17 Tính toán mã hash và so sánh checksum....15
Hình 18 Checkwork....16

DANH MỤC CÁC BẢNG BIỂU

CHƯƠNG 1: GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH

1.1 Giới thiệu chung về bài thực hành

Bài thực hành này tập trung vào kỹ thuật tách tin giấu trong âm thanh bằng cách sử dụng thuật toán nhảy tần – FHSS (Frequency Hopping Spread Spectrum).

Giấu tin trong file âm thanh bằng FHSS (Frequency Hopping Spread Spectrum) là một kỹ thuật mã hóa tín hiệu, nơi thông tin được giấu vào tần số của tín hiệu âm thanh. Cách thực hiện và tách thông tin này có thể được mô tả ngắn gọn như sau:

Cách giấu thông tin:

1. **Chia nhỏ thông tin cần giấu**: Thông tin (ví dụ: văn bản, hình ảnh) sẽ được mã hóa thành các bit.
2. **Tần số nhảy**: Sử dụng FHSS, thông tin được "nhảy" qua các tần số khác nhau trong tín hiệu âm thanh theo một dãy tần số ngẫu nhiên. Dãy tần số này được chọn sao cho nó chỉ có thể được giải mã bởi người nhận có thông tin về cách thức nhảy tần số.
3. **Gửi tín hiệu âm thanh**: Tín hiệu âm thanh chứa thông tin được ẩn giấu thông qua các tần số nhảy.

Cách tách thông tin:

1. **Xác định chu kỳ nhảy tần số**: Người nhận phải biết chính xác cách thức nhảy tần số (giống như người gửi).
2. **Lọc tần số**: Lọc và tái tạo tín hiệu âm thanh bằng cách sử dụng tần số nhảy đã biết, sau đó giải mã thông tin giấu bên trong.
3. **Giải mã**: Sau khi tách ra các phần tín hiệu, thông tin được giải mã từ các bit đã ẩn.

Kỹ thuật này mang lại sự bảo mật cao vì tần số nhảy thay đổi liên tục, khiến việc nhận diện và giải mã trở nên khó khăn nếu không biết chính xác cách thức nhảy tần số.

1.2 Nội dung và hướng dẫn bài thực hành

1.2.1 Mục đích

Giúp sinh viên nắm bắt cách thức tách tin bằng FHSS và cách sử dụng SCP để gửi tệp.

1.2.2 Yêu cầu đối với sinh viên

- Hiểu rõ khái niệm giấu tin và vai trò của giấu tin trong âm thanh.
- Nắm được nguyên lý hoạt động của thuật toán giấu tin Frequency Hopping Spread Spectrum (FHSS).

1.2.3 Nội dung thực hành

Khởi động bài lab, tải bài thực hành bằng imodule:

Imodule imodule.tar

Hoặc giải nén file imodule.tar vào foder trunk/labs
Vào terminal, gõ:

Labtainer -r stego-code-fhss-extract

(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, màn hình sẽ xuất hiện 2 terminal để làm bài thực hành, một terminal đại diện cho máy gửi (sender), một terminal đại diện cho máy nhận(receiver)

- Để phục vụ cho bài thực hành, sinh viên được cung cấp một file code python ***extract.py*** thực hiện tách tin từ file âm thanh ***stego_audio.wav.***
- Kiểm tra địa chỉ IP ở 2 máy.
- Ở máy receiver, sinh viên cài đặt cấu hình SSH như sau:
    - Ở file /etc/ssh/sshd_config, đảm bảo cấu hình mở cổng 22, cho phép truy cập bằng mật khẩu vào đăng nhập bằng root:
        - Port 22
        - PasswordAuthentication yes
        - PermitRootLogin yes
    - Khởi động lại SSH với cấu hình trên: ***sudo systemctl restart ssh.***
- Ở máy sender, gửi file âm thanh bằng lệnh ***scp stego_audio.wav ubuntu@172.33.0.4:/home/ubuntu*** với password là 1.
- Tương tự gửi file checksum chữa mã hash có khóa của file âm thanh.
- Sau khi gửi âm thanh, ở máy receiver, sinh viên tiến hành tách tin bằng lệnh: ***python3 extract***
- Sau khi nhận được tin, tiến hành kiểm tra tin có bị chỉnh sửa không, tính toán mã hash với key nằm trong file key.txt: ***openssl dgst -sha256 -hmac "<điền khóa vào đây>" stego_audio.wav***

- So sánh kết quả vừa tạo ra với nội dung file checksum và kết thúc bài lab.
- Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

  stoplab

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:

labtainer –r stego-code-fhss-extract

CHƯƠNG 2: PHÂN TÍCH YÊU CẦU BÀI THỰC HÀNH

2.1 Thiết kế bài thực hành

Bài lab gồm 2 container là sender và receiver.

Cấu hình Docker

- Bài lab chạy với image là .network2
- Cần cài thêm thư viện numpy, scipy trong dockerfiles của receiver
- Docs lưu lại hướng dẫn thực hành cho sinh viên

Các nhiệm vụ cần thực hiện để thành công:

- Khởi động ssh
- Gửi file âm thanh và checksum đến receiver
- Tách tin từ file âm thanh.
- Tính toán checksum của file âm thanh
- 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 dưới đây:

```
start_ssh = receiver:.bash_history : CONTAINS : systemctl restart ssh
send_file = sender:.bash_history : CONTAINS : ubuntu@172.33.0.4:/home/ubuntu
extracted_message = receiver:*.stdout : CONTAINS : Congrat you got the message!!!
send_checksum = receiver:checksum  : CONTAINS : HMAC-SHA256(stego_audio.wav)=
double_check = receiver:made_checksum : CONTAINS : HMAC-SHA256(stego_audio.wav)=
9c997602a0d7a63e238c5a0e6235949ee73787191d794bc9f6b0266a5a26ca74
```

Hình 1 File result.config

start_ssh : Khởi động ssh ở máy receiver

send_file : Kiểm tra đã gửi file âm thanh từ máy sender đến máy receiver hay chưa

extracted_message: Kiểm tra đã tách tin từ file âm thanh hay chưa

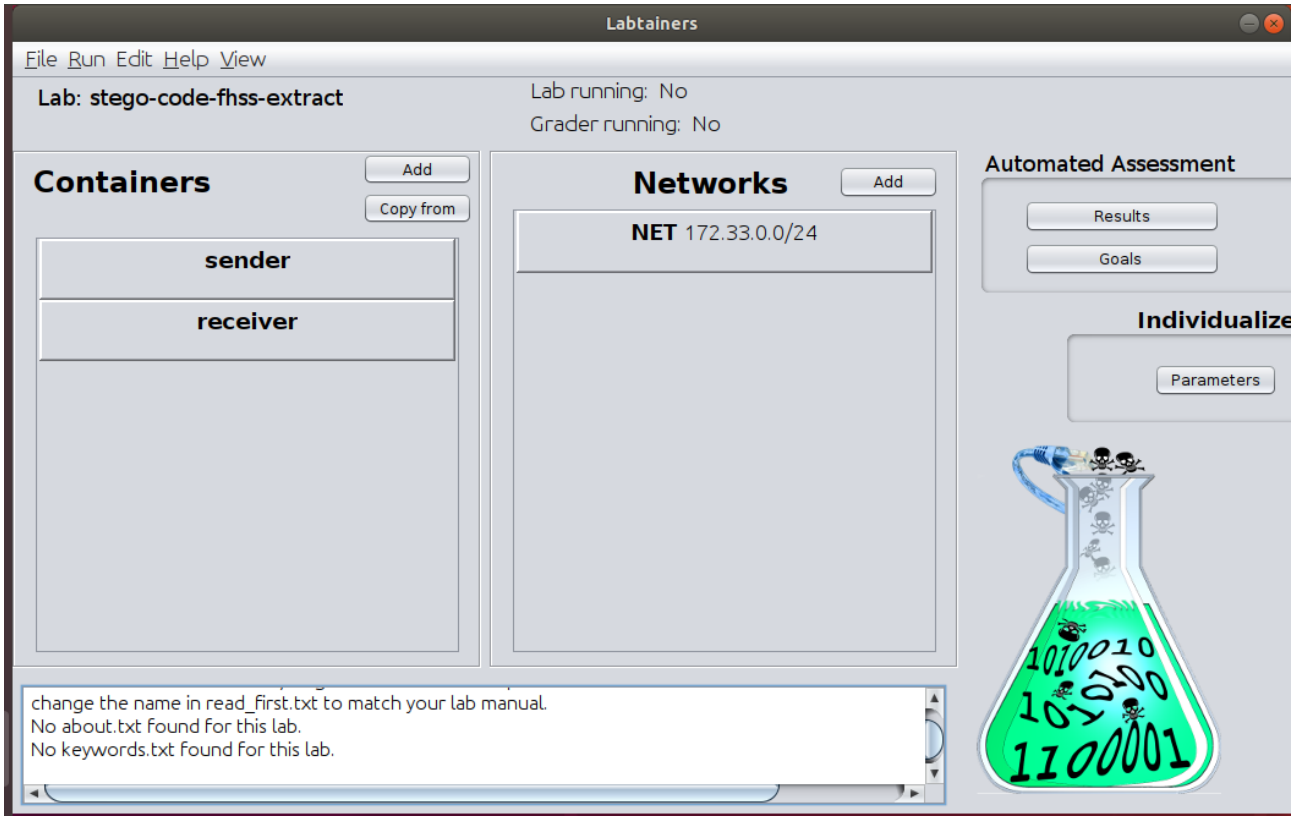
send_checksum: Kiểm tra máy receiver đã nhận được file checksum

hay chưa

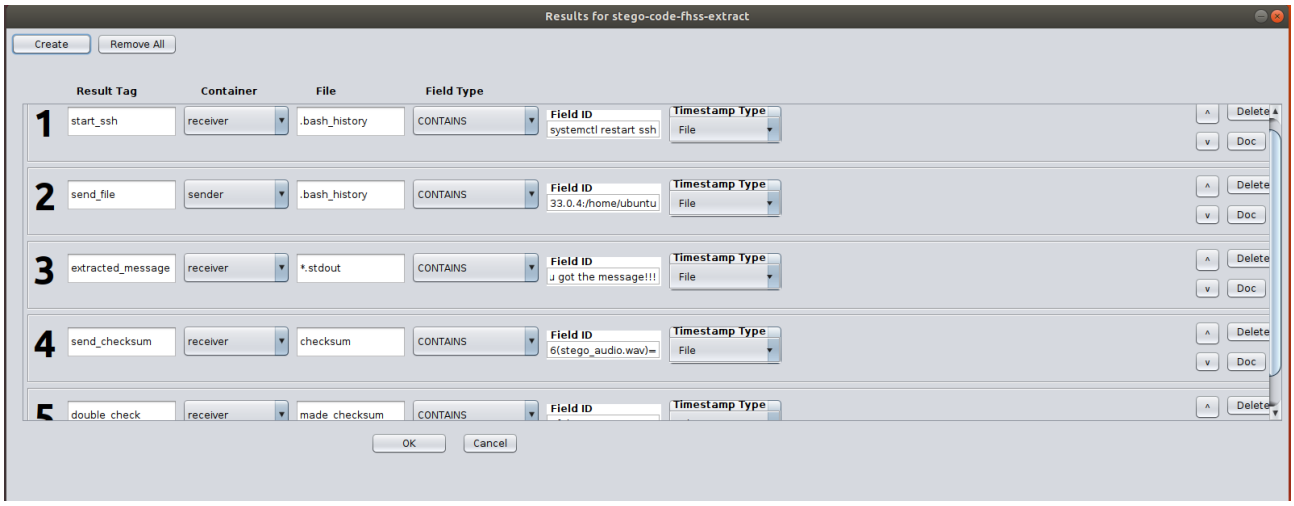
double_check: Kiểm tra xem đã tín toán mã hash của file âm thanh

hay chưa

2.2 Cài đặt và cấu hình máy ảo



Hình 2 Giao diện labedit



Hình 3 Giao diện result.config

2.3 Tích hợp và triển khai

Bài thực hành được tiển khai như sau:

Docker

Đường dẫn: Docker Hub

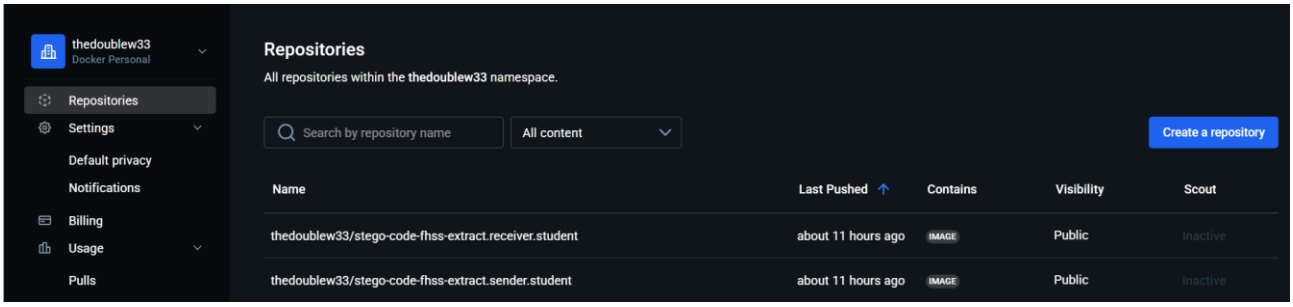
Thêm registry cho bài thực hành

Truy cập vào thư mục trunk/distrib gõ lệnh: *docker login* đăng nhập tài khoản DockerHub

Sử dụng lệnh *./publish.py -d -l stego-code-fhss-extract* để đẩy images của bài thực hành lên DockerHub

```
student@ubuntu:~/labtainer/trunk/distrib$ ./publish.py -d -l stego-code-fhss-extract
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]
No images for stego-code-fhss-extract
```

Hình 4 Đẩy image bài thực hành lên docker hub



Hình 5 Image trên dockerhub

Github

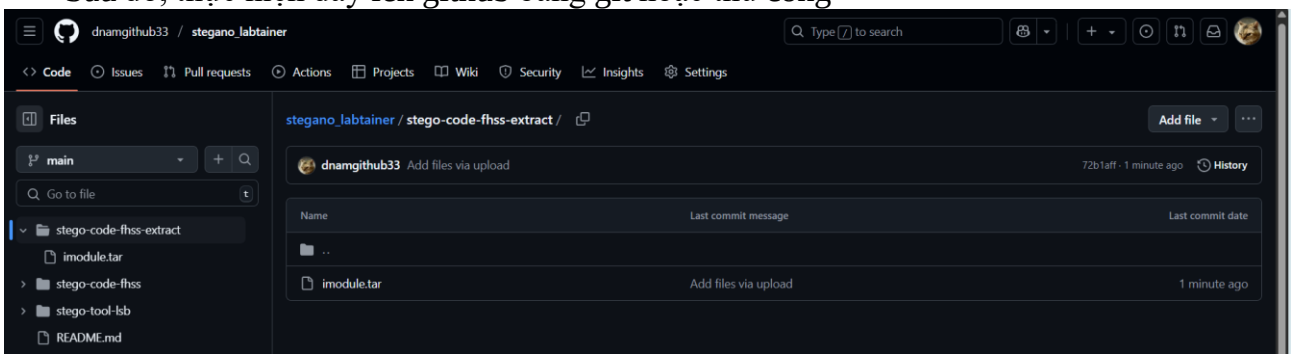
Đường dẫn: tại đây

Ở đường dẫn $LABTAINER_DIR/distrib, tạo file tar bằng create-imodules.sh

```
student@ubuntu:~/labtainer/trunk/scripts/designer/bin$ ./create-imodules.sh
lab is stego-code-fhss-extract
Do docs
******************************************************
**  Post /home/student/labtainer/trunk/imodule.tar to your web server **
******************************************************
student@ubuntu:~/labtainer/trunk/scripts/designer/bin$
```

Hình 6 Tạo file imodule

Sau đó, thực hiện đẩy lên github bằng git hoặc thủ công

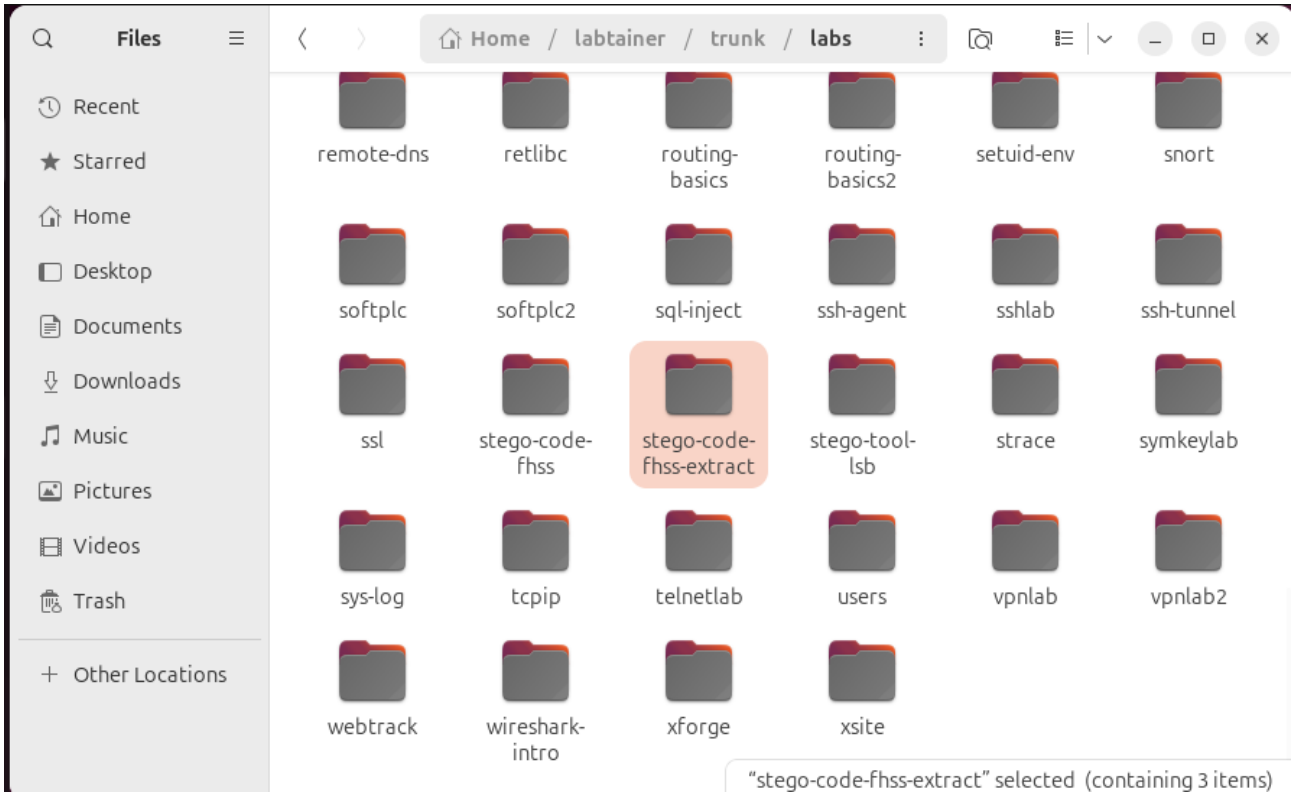


Hình 7 File imodule trên github

CHƯƠNG 3: 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:

Khởi động bài lab, tải bài thực hành bằng imodule, trong trường hợp không dùng được imodule, giải nén file thực hành trong folder labtainer/trunk/labs:



Hình 8 Cài đặt bài thực hành

Vào terminal, gõ:

Labtainer -r stego-code-fhss-extract

(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)

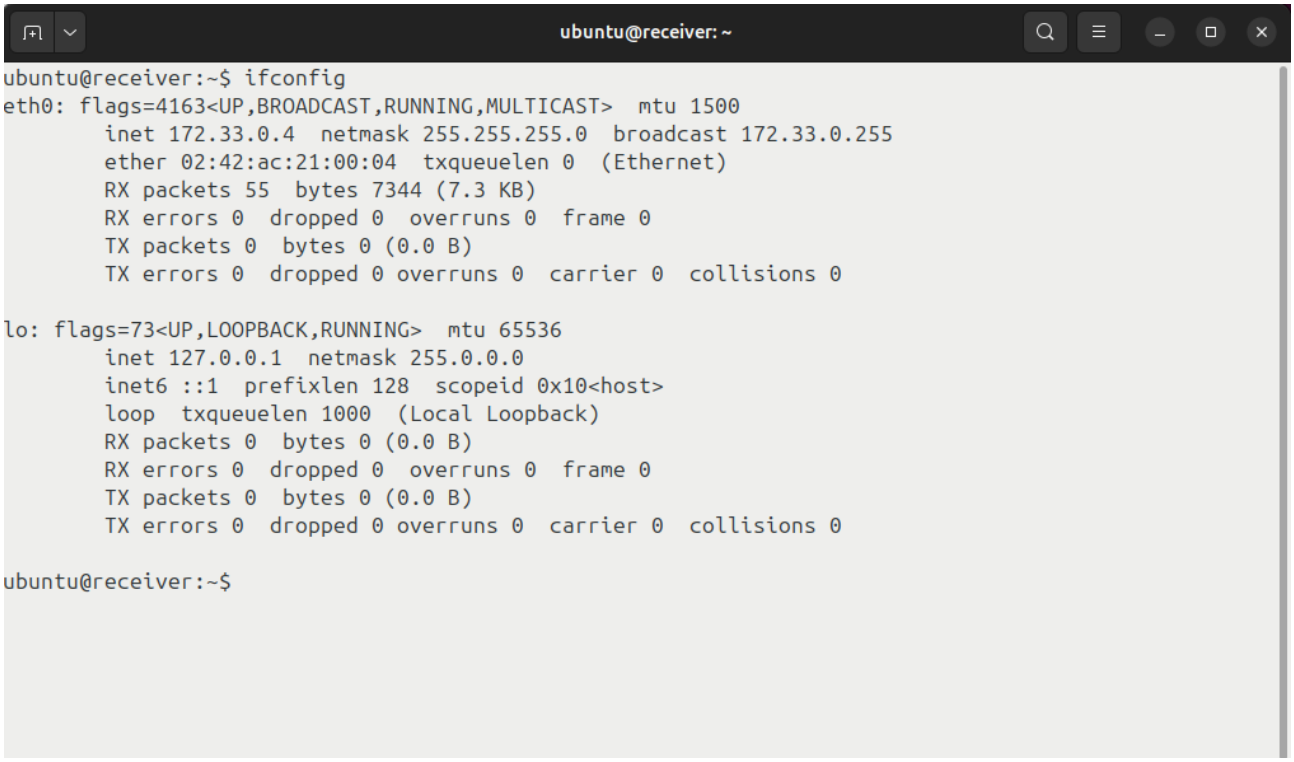
```
student@LabtainerVMware:~/labtainer/labtainer-student$ labtainer stego-code-fhss-extract
latest: Pulling from thedoublew33/stego-code-fhss-extract.sender.student
bade6855aaf4: Pull complete
4a90ec3a29f7: Pull complete
8e7e24552aa8: Pull complete
feb32af91de5: Pull complete
a62f2ca25ef9: Pull complete
978402a04cb0: Pull complete
1b7285d549f4: Pull complete
79617e670760: Pull complete
233b36c94357: Pull complete
6b5a151afeb1: Pull complete
Digest: sha256:88f2397559a3d563b369e48870424a727a199a6e258eb58779aa671edec8c9e0
Status: Downloaded newer image for thedoublew33/stego-code-fhss-extract.sender.student:latest
latest: Pulling from thedoublew33/stego-code-fhss-extract.receiver.student
5249fa028ec6: Pull complete
ea13a6cc2d08: Pull complete
8e7e24552aa8: Pull complete
7c5466e4d6da: Pull complete
70d7019069c2: Pull complete
715f2bf8d60d: Pull complete
1899434cba9b: Pull complete
1f1ca67a0450: Pull complete
355d498ecc53: Pull complete
09d199448abb: Pull complete
962f0bc1e1cc: Pull complete
6b5a151afeb1: Pull complete
Digest: sha256:d311ab371e41e2999435247014c73b6a5916489ddd0db203734c89107a15351a
Status: Downloaded newer image for thedoublew33/stego-code-fhss-extract.receiver.student:latest
non-network local connections being added to access control list
```

Hình 9 Khởi động badi lab

- Kiểm tra địa chỉ IP ở 2 máy.

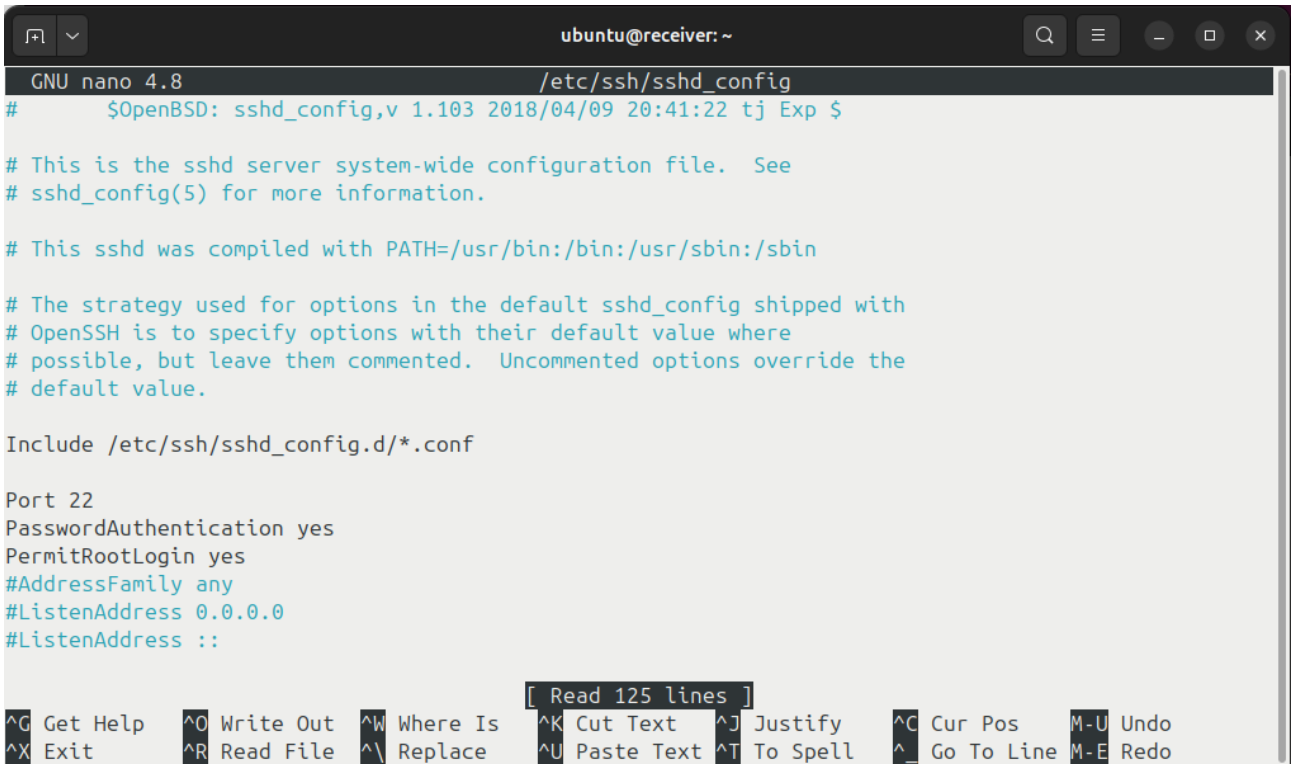
```
ubuntu@sender:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.33.0.3  netmask 255.255.255.0  broadcast 172.33.0.255
        ether 02:42:ac:21:00:03  txqueuelen 0  (Ethernet)
        RX packets 49  bytes 6776 (6.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ubuntu@sender:~$
```

Hình 10 Địa chỉ IP máy sender



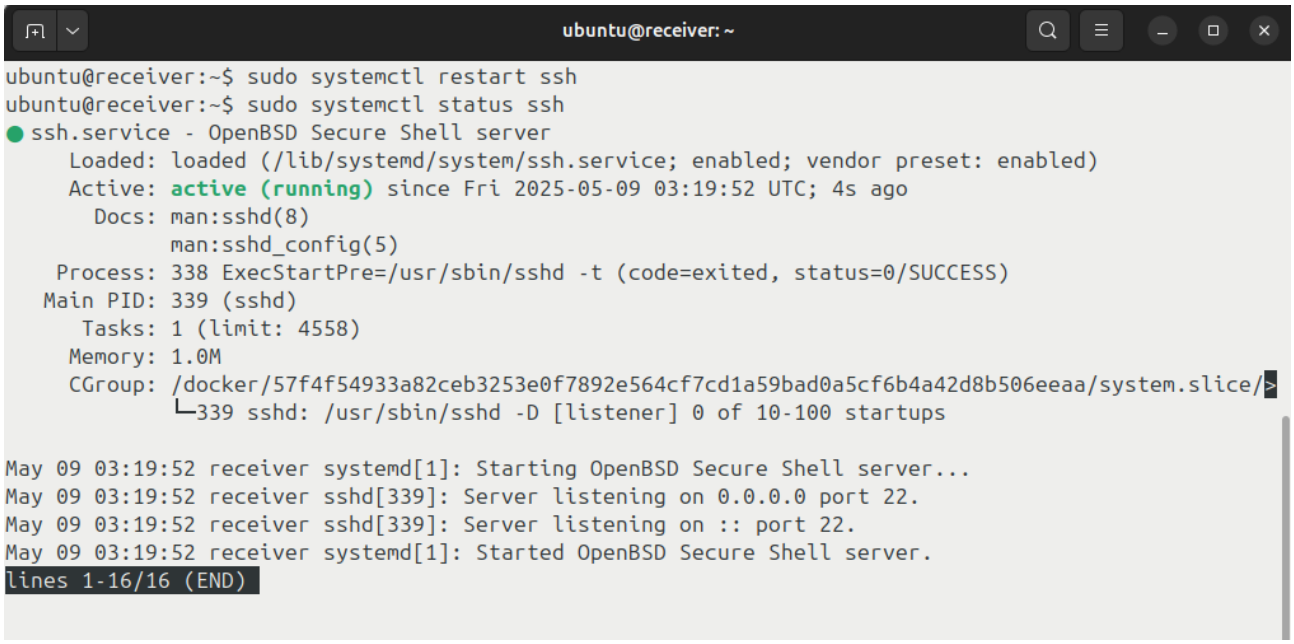
Hình 11 Địa chỉ IP máy receiver

- Ở máy receiver, sinh viên cài đặt cấu hình SSH như sau:
    - Ở file /etc/ssh/sshd_config, đảm bảo cấu hình mở cổng 22, cho phép truy cập bằng mật khẩu vào đăng nhập bằng root:
        - Port 22
        - PasswordAuthentication yes
        - PermitRootLogin yes



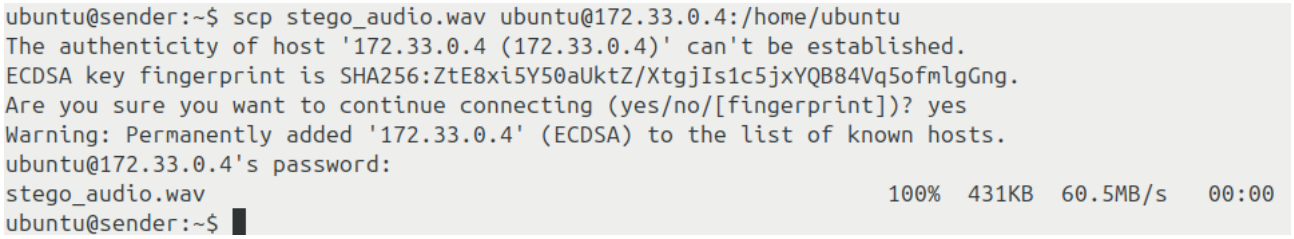
Hình 12 Cấu hình SSH máy receiver

- Khởi động lại SSH với cấu hình trên: ***sudo systemctl restart ssh.***



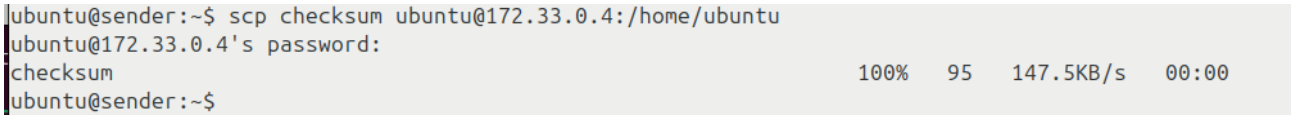
Hình 13 Khởi động lại SSH

- Ở máy sender, gửi file âm thanh bằng lệnh ***scp stego_audio.wav ubuntu@172.33.0.4:/home/ubuntu*** với password là 1.



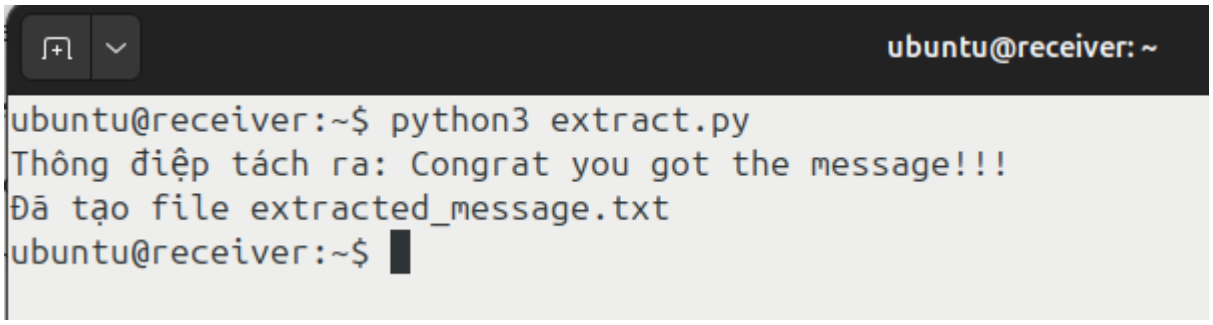
Hình 14 Gửi file âm thanh sang receiver

- Tương tự gửi file checksum chữa mã hash có khóa của file âm thanh.



Hình 15 Gửi file checksum sang receiver

- Sau khi gửi âm thanh, ở máy receiver, sinh viên tiến hành tách tin bằng lệnh: ***python3 extract***



Hình 16 Tách tin từ file âm thanh

- Sau khi nhận được tin, tiến hành kiểm tra tin có bị chỉnh sửa không, tính toán mã hash với key nằm trong file key.txt: ***openssl dgst -sha256 -hmac "<điền khóa vào đây>" stego_audio.wav > made_checksum***
- So sánh kết quả vừa tạo ra với nội dung file checksum và kết thúc bài lab.

```
ubuntu@receiver:~$ openssl dgst -sha256 -hmac $(cat key.txt) stego_audio.wav > made_checksum
ubuntu@receiver:~$ cat checksum
HMAC-SHA256(stego_audio.wav)= 9c997602a0d7a63e238c5a0e6235949ee73787191d794bc9f6b0266a5a26ca74
ubuntu@receiver:~$ cat made_checksum
HMAC-SHA256(stego_audio.wav)= 9c997602a0d7a63e238c5a0e6235949ee73787191d794bc9f6b0266a5a26ca74
ubuntu@receiver:~$
```

Hình 17 Tính toán mã hash và so sánh checksum

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

  stoplab

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.

Kết quả checkwork

```
student@LabtainerVMware:~/labtainer/labtainer-student$ checkwork
Results stored in directory: /home/student/labtainer_xfer/stego-code-fhss-extract
Successfully copied 332kB to stego-code-fhss-extract-igrader:/home/instructor/b21dcat135.stego-code-fhss-extract.lab
Successfully copied 2.05kB to /home/student/labtainer_xfer/stego-code-fhss-extract
Labname stego-code-fhss-extract

Student              |       start_ssh |       send_file | extracted_messa |   send_checksum |    double_check |
==================== | =============== | =============== | =============== | =============== | =============== |
b21dcat135           |               Y |               Y |               Y |               Y |               Y |
What is automatically assessed for this lab:
```

Hình 18 Checkwork

Khởi động lại bài lab:

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

labtainer –r stego-code-fhss-extract

TÀI LIỆU THAM KHẢO

[1] Bài giảng Các kỹ thuật giấu tin, PGS. TS Đỗ Xuân Chợ