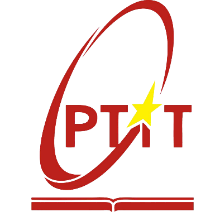
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA AN TOÀN THÔNG TIN**

-------🙞🙜🕮🙞🙜-------



**CÁC KỸ THUẬT GIẤU TIN**

**Đề bài: Giấu tin bằng phương pháp tạo NALU giả trong video H264**

|  |  |
| --- | --- |
| **HỌ TÊN:** | **PHẠM HẢI DƯƠNG** |
| **MÃ SINH VIÊN:** | **B21DCAT072** |

Giảng viên hướng dẫn: PGS.TS. Đỗ Xuân Chợ

**HÀ NỘI, 04/2025**

**MỤC LỤC**

[1. Giới thiệu chung về bài thực hành 5](#_Toc196590796)

[2. Nội dung và hướng dẫn bài thực hành 5](#_Toc196590797)

[1.1. Mục đích 5](#_Toc196590798)

[1.2. Yêu cầu đối với sinh viên 6](#_Toc196590799)

[1.3. Nội dung thực hành 6](#_Toc196590800)

[3. Phân tích yêu cầu của bài thực hành 8](#_Toc196590801)

[4. Cài đặt và cấu hình máy ảo 9](#_Toc196590802)

[5. Tích hợp và triển khai 14](#_Toc196590803)

[5.1. Docker Hub 14](#_Toc196590804)

[5.2. GitHub 14](#_Toc196590805)

[6. Thử nghiệm và đánh giá 15](#_Toc196590806)

**DANH MỤC HÌNH ẢNH**

[Hình 1: Giao diện Labedit 9](#_Toc196590811)

[Hình 2: Cấu hình result 10](#_Toc196590812)

[Hình 3: Cấu hình Result 10](#_Toc196590813)

[Hình 4: Cấu hình Goal 11](#_Toc196590814)

[Hình 5: Docker container bob 11](#_Toc196590815)

[Hình 6: Docker container bob 12](#_Toc196590816)

[Hình 7: Docker container alice 13](#_Toc196590817)

[Hình 8:Docker container alice 13](#_Toc196590818)

[Hình 9: Docker 14](#_Toc196590819)

[Hình 10: Đẩy lab lên github 15](#_Toc196590820)

# **Giới thiệu chung về bài thực hành**

Bài thực hành này được thiết kế nhằm giúp sinh viên nắm bắt kỹ thuật ẩn dữ liệu trong luồng video H.264-AVC thông qua việc chèn các NAL Unit không hợp chuẩn (Fake NALU). Thông qua quá trình xây dựng, chèn và giải mã các NAL Unit đặc biệt có nal\_unit\_type = 31, sinh viên sẽ được thực hành trực tiếp trên luồng bit video, hiểu sâu về cấu trúc của NALU và cách thức video được mã hóa, đóng gói và truyền tải.

Chuẩn mã hóa H.264-AVC phân chia luồng video thành các đơn vị gọi là NALU (Network Abstraction Layer Unit), trong đó mỗi NAL có một type định nghĩa chức năng cụ thể. Trong bài này, sinh viên sẽ khai thác vùng type 31 – vốn được đánh dấu là unspecified trong chuẩn – để giấu thông tin một cách không chính thống nhưng hiệu quả. Kỹ thuật này cho phép nhúng dữ liệu vào video mà không làm thay đổi hình ảnh hoặc cấu trúc logic của nội dung, đồng thời có thể tồn tại ẩn nếu không bị giải mã hoặc phân tích kỹ.

Sinh viên sẽ thực hiện toàn bộ quy trình: tạo luồng video raw định dạng .h264, chèn dữ liệu bí mật vào giữa các NALU bằng script Python, đóng gói lại video sang định dạng .mp4 để truyền tải qua mạng, sau đó giải nén và trích xuất lại dữ liệu từ fake NALU trên máy đích. Quá trình này giúp rèn luyện kỹ năng xử lý luồng video thô, sử dụng các công cụ như FFmpeg, MP4Box và Python, đồng thời hiểu rõ các giới hạn và rủi ro khi sử dụng NALU không hợp chuẩn trong các định dạng video phổ biến.

Sau khi hoàn thành bài thực hành, sinh viên sẽ có cái nhìn toàn diện về kỹ thuật giấu dữ liệu bằng cách thao túng NALU, nắm rõ sự khác biệt giữa luồng .h264 và định dạng .mp4, cũng như hiểu rõ các yếu tố kỹ thuật ảnh hưởng đến tính bền vững và khả năng khôi phục dữ liệu sau khi truyền hoặc chuyển đổi định dạng. Đây là nền tảng quan trọng để sinh viên tiếp cận các kỹ thuật steganography nâng cao trong miền nén video.

# **Nội dung và hướng dẫn bài thực hành**

## **Mục đích**

* Hiểu tổng quan về quy trình mã hóa và giải mã video theo chuẩn H.264-AVC.
* Nắm vững cấu trúc của luồng dữ liệu H.264, đặc biệt là các thành phần NALU (Network Abstraction Layer Unit).
* Áp dụng kỹ thuật giấu tin bằng cách chèn dữ liệu vào các NAL Unit không chuẩn (Fake NALU – nal\_unit\_type = 31) để tránh ảnh hưởng đến nội dung hiển thị.
* Thực hành đóng gói video H.264 thành định dạng .mp4 và phục hồi lại .h264, đảm bảo dữ liệu giấu được bảo toàn trong quá trình truyền tải và xử lý.

## **Yêu cầu đối với sinh viên**

* Có kiến thức cơ bản về kỹ thuật giấu tin trong đa phương tiện (steganography), đặc biệt là giấu tin trong ảnh và video.
* Hiểu rõ khái niệm, cấu trúc NALU và vai trò của từng loại NAL trong chuẩn video H.264-AVC.
* Thành thạo việc sử dụng các công cụ xử lý media như FFmpeg, MP4Box và các script Python để thao tác với luồng video .h264.

## **Nội dung thực hành**

Sinh viên khởi động bài lab

Chạy lệnh:

*labtainer -r stego\_h264\_video\_fake\_nalu*

*(Chú ý: sinh viên sử dụng <TÊN\_TÀI\_KHOẢN> của mình để nhập thông tin người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm.)*

Sau khi khởi động bài lab,có 2 container hiện lên là bob và alice. Sinh viên thực hiện làm lab theo yêu cầu trên 2 container này

***TASK 1: Mã hóa video chuẩn h264 và chuyển dổi thành luồng h264 thô***

* Sau khi khởi động xong một terminal ảo sẽ xuất hiện, terminal sử dụng để thực hành bài lab
* Trong terminal alice đã có sẵn 1 video theo chuẩn mã hóa h264.
* Sinh viên chuyển đổi video đó thành dạng luồng h264thô (raw h264) để phục vụ cho các mục tiêu trong các task tới: ***ffmpeg -i video\_h264.mp4 -c:v copy -bsf:v h264\_mp4toannexb raw.h264***

***TASK 2: Giấu thông tin bằn phương pháp tạo NALU giả***

* Trong chuẩn nén video H.264/AVC, toàn bộ dữ liệu video được tổ chức dưới dạng các đơn vị gọi là NALU (Network Abstraction Layer Unit). Mỗi NALU là một khối dữ liệu chứa header (tiêu đề) và payload (dữ liệu). Header sẽ chỉ định loại của NALU thông qua trường nal\_unit\_type (5 bit), từ đó xác định chức năng của khối dữ liệu này.
* Một số loại NALU phổ biến gồm:
  + 1: Non-IDR slice (khung hình P, B)
  + 5: IDR slice (khung hình I)
  + 6: SEI (Supplemental Enhancement Information)
  + 7: SPS (Sequence Parameter Set)
  + 8: PPS (Picture Parameter Set)
* Theo đặc tả chuẩn, các giá trị nal\_unit\_type từ 24 đến 31 được đánh dấu là "unspecified", tức là không được định nghĩa cụ thể trong chuẩn H.264.
* Chính vùng unspecified NAL types (24–31) là cơ sở kỹ thuật để ta có thể chèn thêm các NALU tùy ý mà không làm sai chuẩn về mặt cú pháp. Trong bài thực hành này, ta chọn nal\_unit\_type = 31, tạo thành một Fake NALU chứa dữ liệu bí mật. Việc chèn Fake NALU giữa các NALU hợp lệ (ví dụ sau SPS hoặc trước IDR) là một cách hiệu quả để nhúng dữ liệu mà không ảnh hưởng đến khả năng giải mã hay phát lại video.
* Script python insert\_fake\_nalu.py đax được viết sẵn trên máy alice để sinh viên thực hiện giấu tin vào trong nó. Tuy nhiên sinh viên cầ xem code và chỉnh sửa tthông điệp giấu tin (bài thực hành này yêu cầu sinh viên điền mã sinh viên của mình làm thông điệp cần giấu)
* Kết quả sau khi giấu thông điệp bằng code python là tạo ra được file modified.h264, là 1 luồng thô h264 chứa thông điệp đã giấu

***TASK 3: Thử giải mã bằng công cụ ffmpeg***

* Nếu bạn đã làm bài lab về giấu tin trong SEI của video thì chung ta đều biết là sử dụng công cụ ffmpeg có thể đóng gói lại raw h264. Ở bài này chúng ta cũng sẽ thử tương tự: ***ffmpeg -f h264 -i modified.h264 -c:v copy ffmpeg\_output.mp4***
* Tiến hành xem video mới xem có gì lỗi hoặc có nhận ra gì bất thường bằng mắt thường không: ***ffplay -autoexit ffmpeg\_output.mp4***
* Kiểm tra startú dịch vụ ssh trên máy alice và máy bob, sau đó dùng scp để gửi file video mp4 qua cho bob (mật khẩu là ubuntu)

***scp ffmpeg\_output.mp4 ubuntu@192.168.130.20:/home/ubuntu/***

* Chuyển sang máy bob khi đã nhận được video ffmpeg\_output.mp4. Cũng dùng ffmpeg để chuyển sang dạng raw file và giải mã xem có nhận được thông điệp hay không ?

***ffmpeg -i ffmpeg\_output.mp4 -c:v copy -bsf:v h264\_mp4toannexb modified.h264***

***python3 extract\_fake\_nalu.py***

* Ta nhận thấy rằng bob không thể giải mã ra thông điệp mà alice giấu. Đây không phải là lỗi vì đơn giản là chúng ta tạo fake nalu chưng thông điệp giấu với nal\_unit\_type = 31. Đây không phải là 1 NALU TYPE được định nghĩa chuẩn trong h264. Vì vậy nên ffmpeg khi đóng gói từ dạng video thô (raw h264) sang dạng video chuẩn (mp4) sẽ loại bỏ tất cả các NAL Unit không chuẩn, nên rõ ràng là thông điệp giấu đã bị mất đi.

***TASK 4: Giãi mã thông điệp***

* Như vậy rõ ràng là ta không thể dùng công cụ ffmpeg để đóng gói luồng h264 thô thành video, nên ta phải sử dụng 1 công cụ khác để làm điều đó mà vẫn giữ lại được fake nalu mà alice sử dụng để giấu tin.
* Công cụ MP4Box đã được cài đặt sẵn trên máy của alice, có thể kiểm tra bằng câu lệnh: ***MP4Box -version***
* Alice dùng công cụ này để gói lại luồng h264 thô thành video mp4: ***MP4Box -add modified.h264 -new mp4box\_output.mp4.*** Ta sẽ thấy các cảnh báo từ MP4Box, báo rằng nó không xử lý đặc biệt NAL type 31 (Fake NALU), nhưng vẫn chấp nhận và thêm nó vào video mp4. Điều này giúp cho nội dung giấu vẫn giữ nguyên
* Alice dùng scp chuyển file video mp4 mới cho bob.
* Bob xem video và không nhận thấy gì khác thường: ***ffplay -autoexit mp4box\_output.mp4***

Bob cũng sử dụng MP4Box để extract video thành dạng raw h264: ***MP4Box -raw 1 mp4box\_output.mp4***

* Bob sử dụng script python để tách ra thông điệp (tuy nhiên cần sửa đổi tên file trong code extract\_fake\_nalu.py thành tên file raw h264 được tạo ra bởi MP4Box): ***python3 extract\_fake\_nalu.py***

Kết thúc lab:

* Trên terminal khởi động lab, sinh viên sử dụng lệnh:

***Stoplab***

* Khi bài lab kết thúc, một tệp lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab. Sinh viên cần nộp file .lab để chấm điểm.
* Để kiểm tra kết quả khi trong khi làm bài thực hành sử dụng lệnh:

***checkwork <tên bài thực hành>***

* Sinh viên cần nộp file .lab để chấm điểm.
* Kiểm tra kết quả trong quá trình làm bài:

***checkwork <tên bài lab>***

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

***labtainer -r stego\_h264\_video\_fake\_nalu***

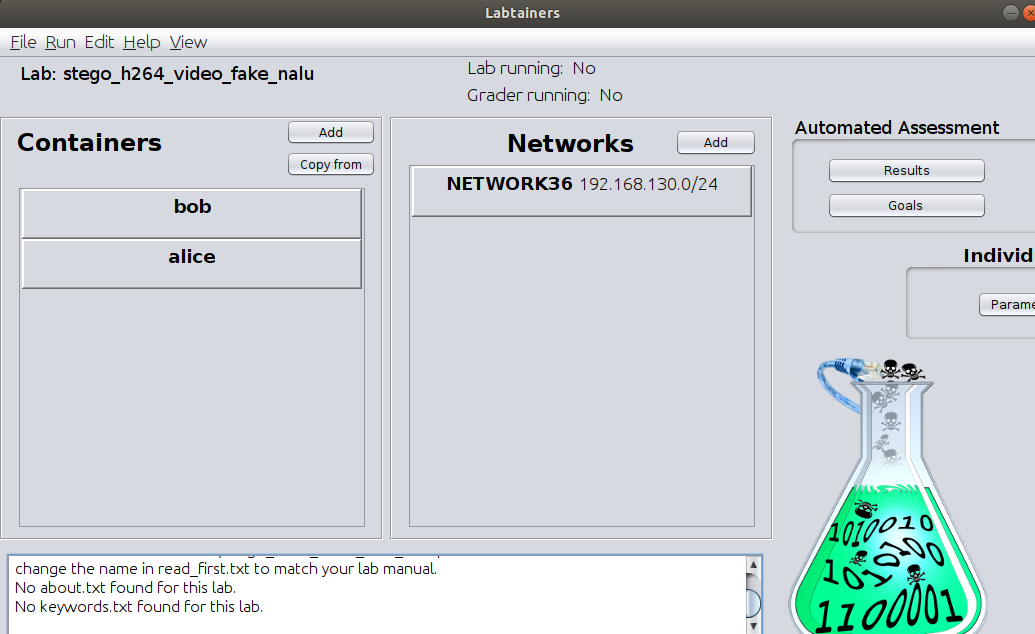
# **Phân tích yêu cầu của bài thực hành**

Bài thực hành được triển khai trên hai máy ảo trong môi trường Labtainer, bao gồm máy alice và máy bob, trong đó:

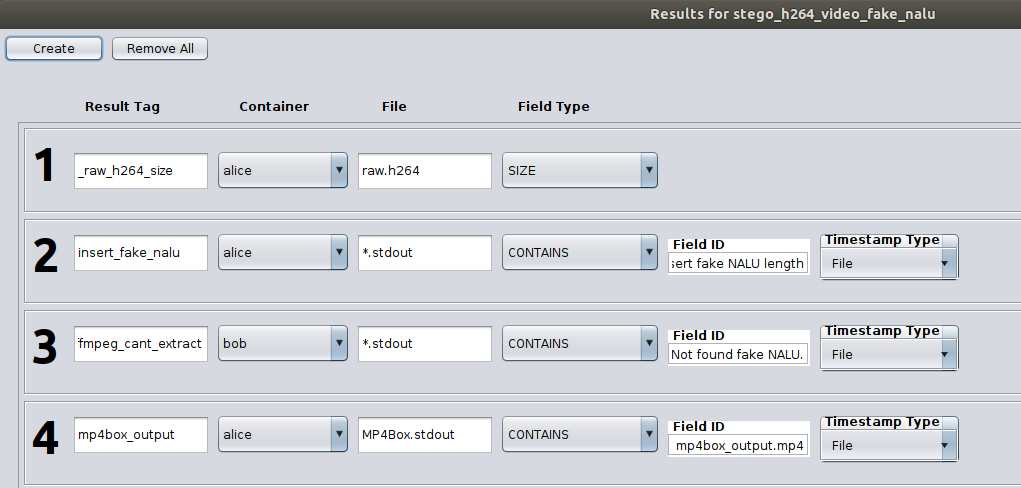
* alice là máy gửi, có chứa một video mẫu ở dạng chuẩn H.264 (Annex B).
* bob là máy nhận, dùng để tiếp nhận video chứa dữ liệu giấu và thực hiện giải mã, trích xuất thông điệp ẩn bên trong.

Sinh viên cần phần tích được cấu trúc luồng h264, tìm vị trí phù hợp và tạo được fake nalu chứa thông tin cần giấu và giấu vào trong video, đồng thời biết cách sử dụng công cụ để không làm mất mát thông tin giấu. Sau đó, sử dụng sccript Python tiến hành giải mã và khôi phụn thông điệp ban đầu.

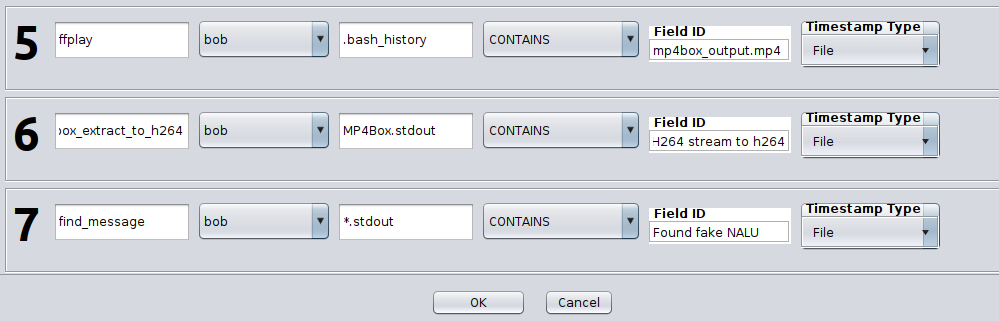
# **Cài đặt và cấu hình máy ảo**



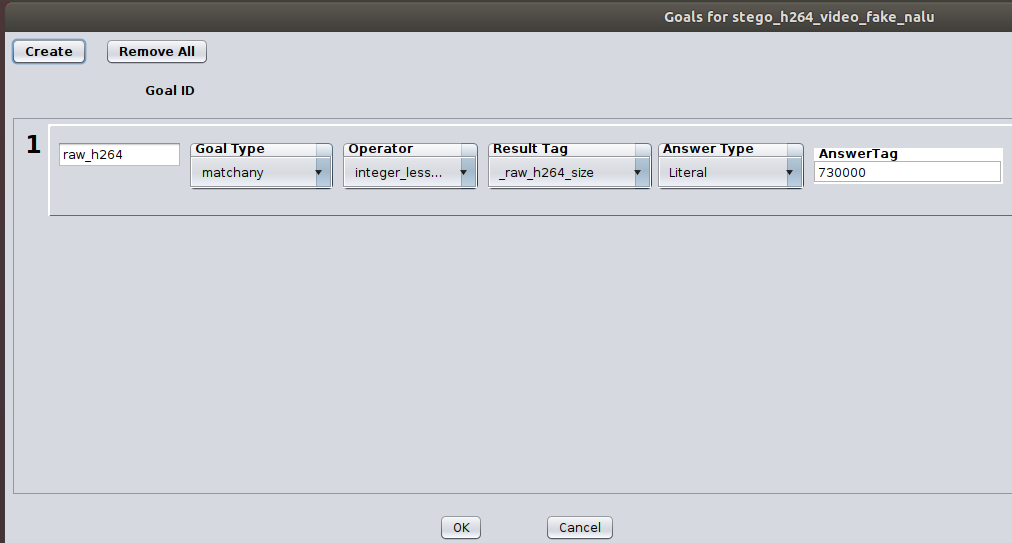
Hình 1: Giao diện Labedit



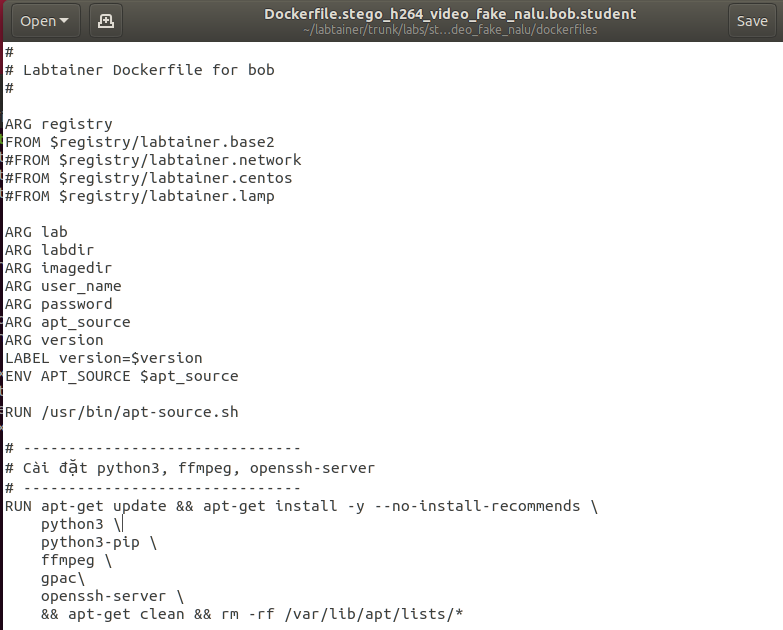
Hình 2: Cấu hình result



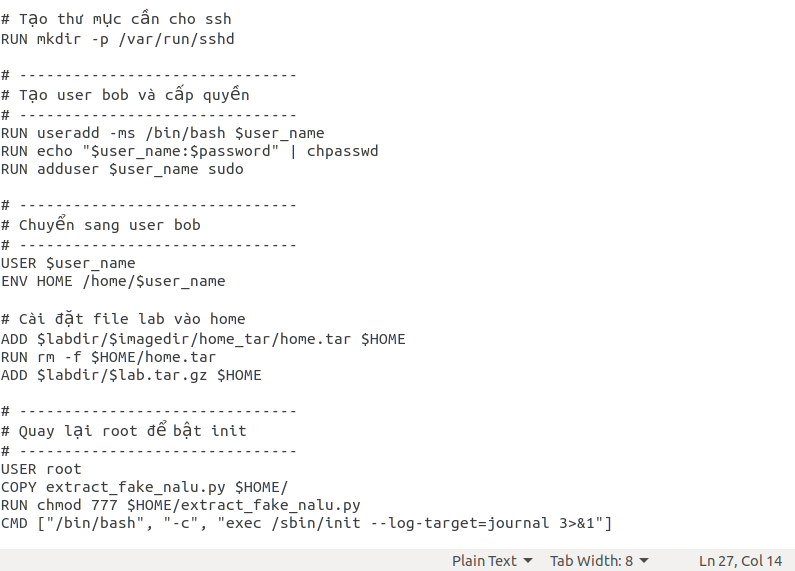
Hình 3: Cấu hình Result



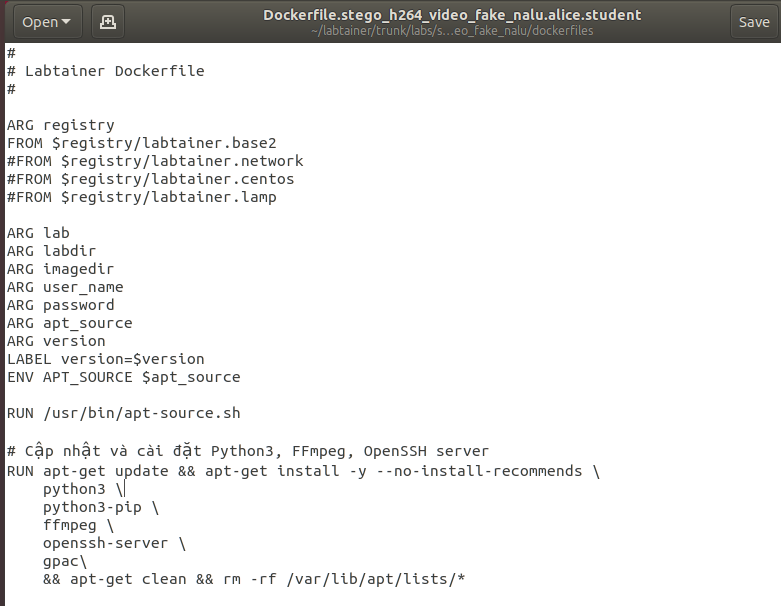
Hình 4: Cấu hình Goal



Hình 5: Docker container bob



Hình 6: Docker container bob



Hình 7: Docker container alice



Hình 8:Docker container alice

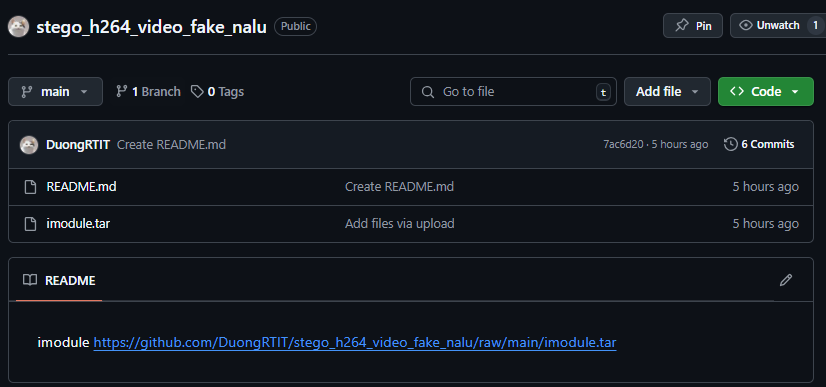
# **Tích hợp và triển khai**

## **Docker Hub**



Hình 9: Docker

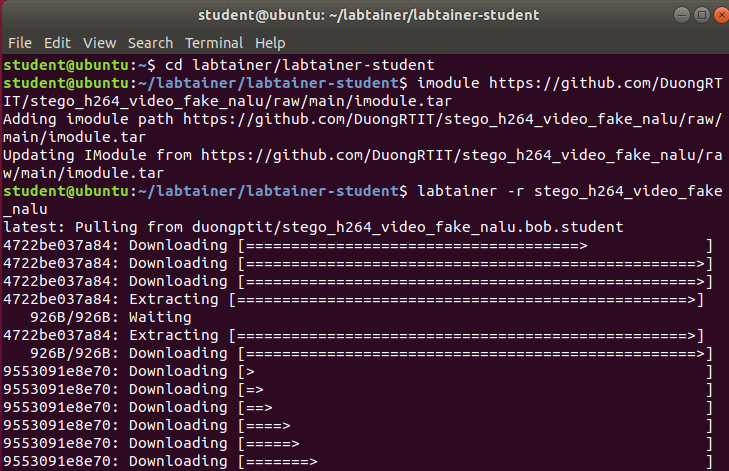
## **Github**

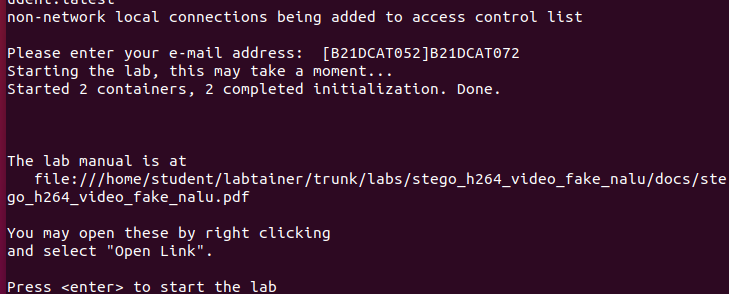


Hình 10: Đẩy lab lên github

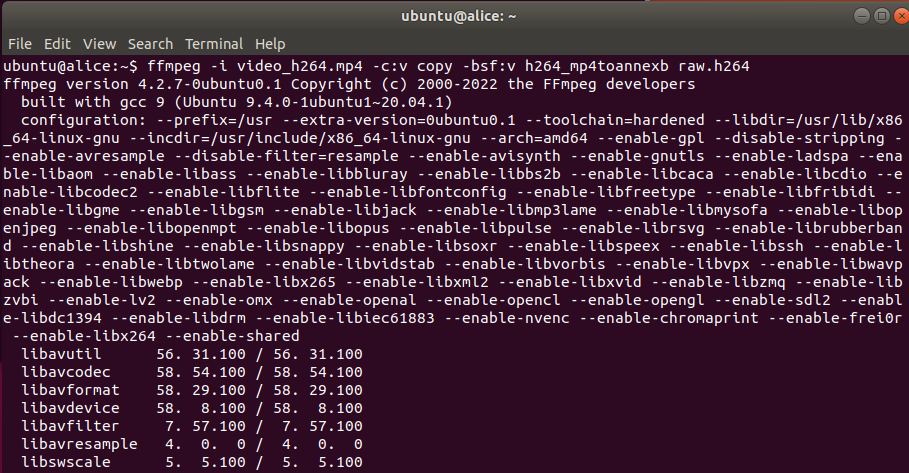
# **Thử nghiệm và đánh giá**

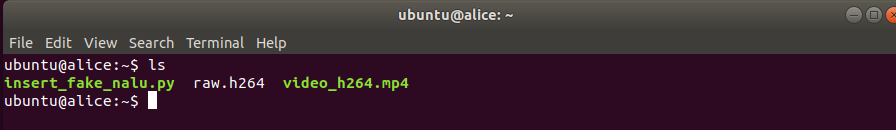
* Tải bài lab về từ Github và chạy bài lab:



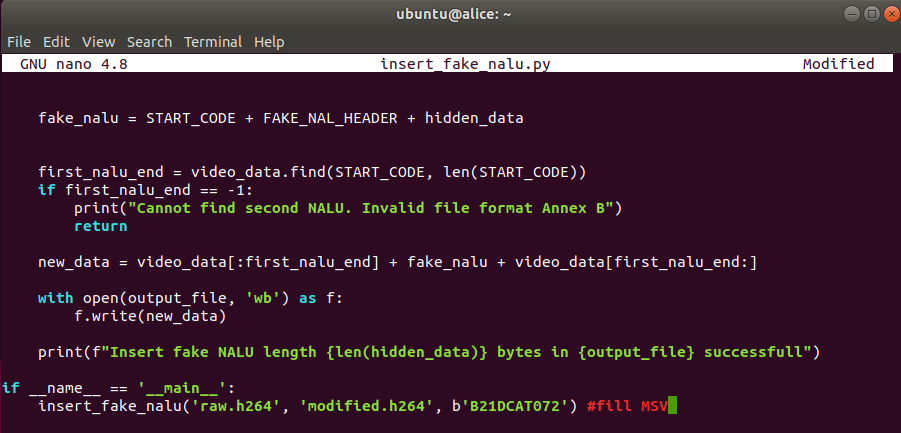


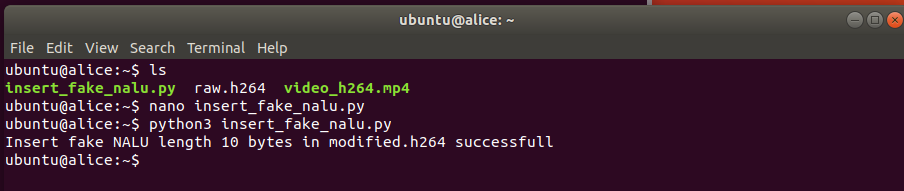
* Thực hiện Task 1:



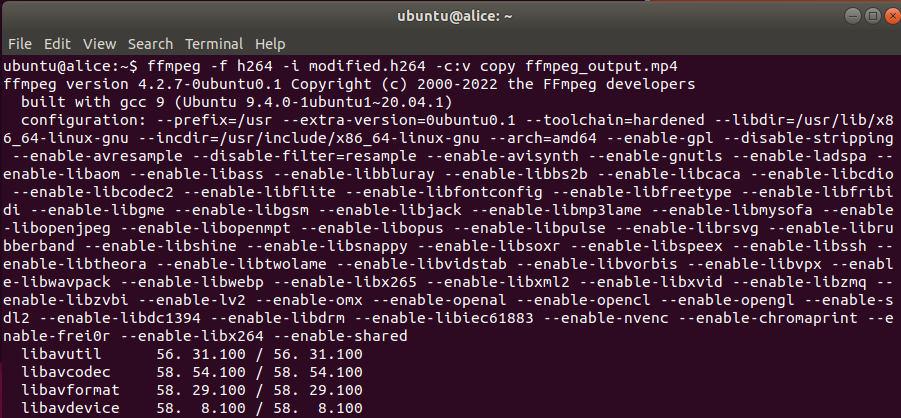


* Thực hiện task 2:

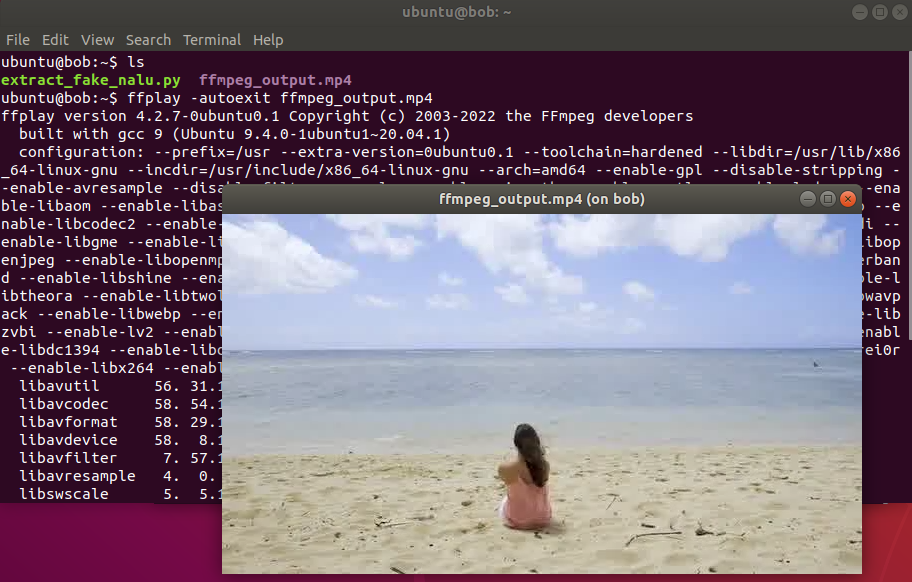


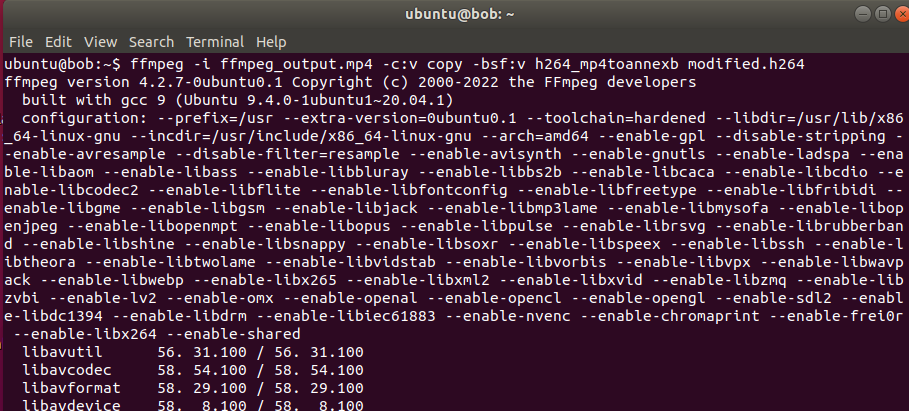


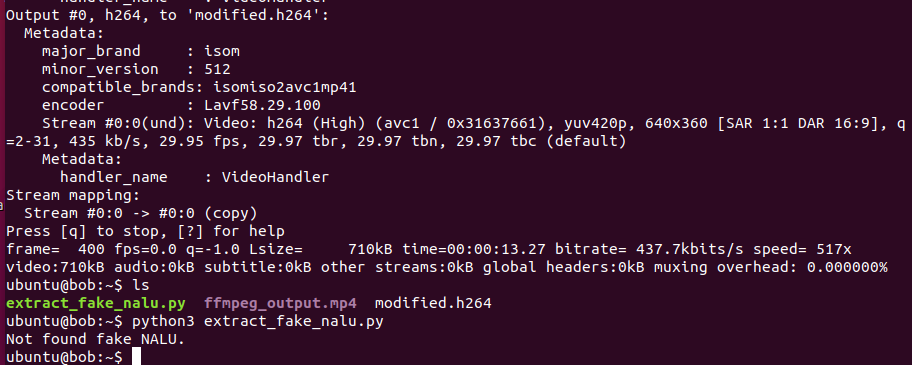
* Thực hiện task 3:



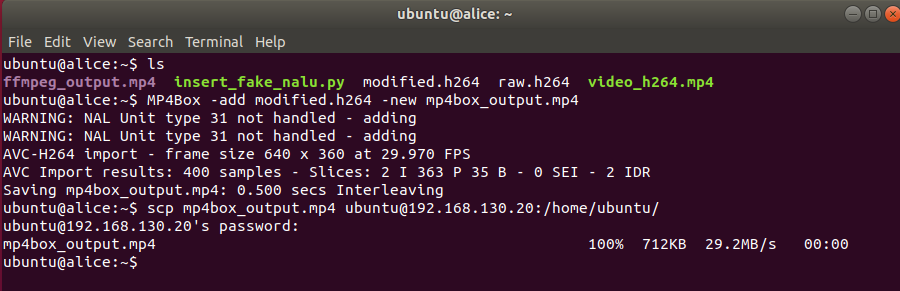


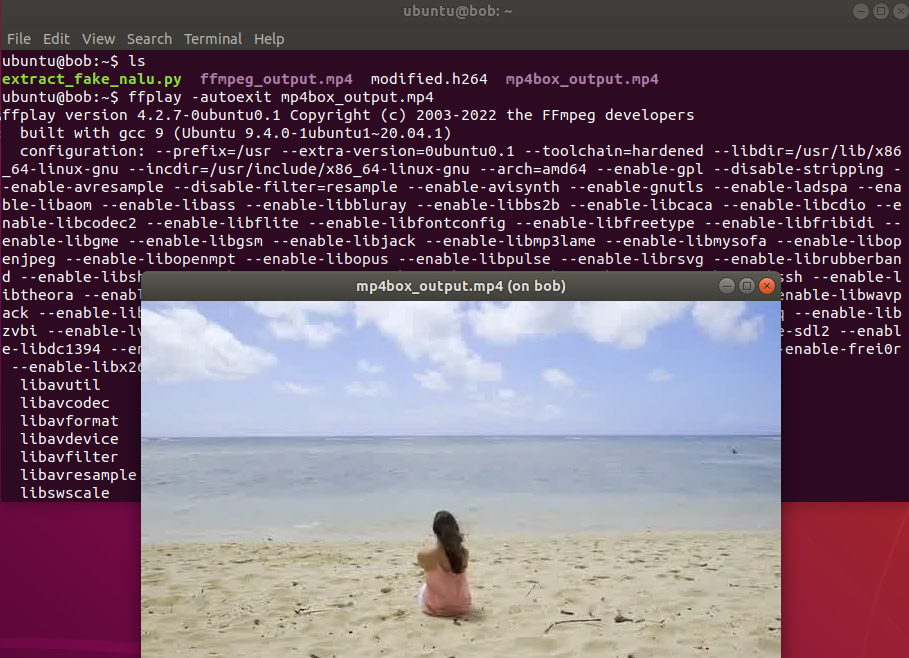


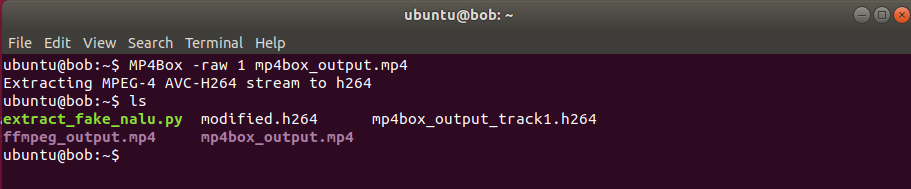


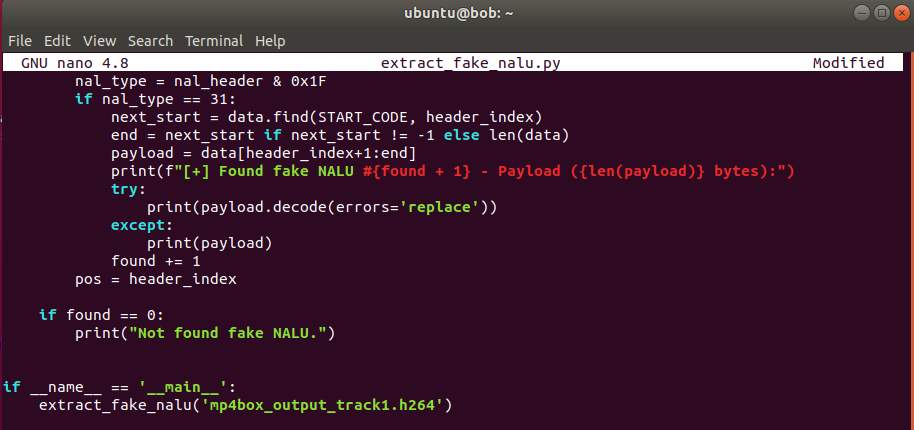


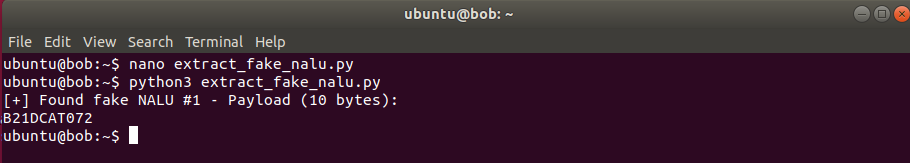
* Thực hiện task 4:











* Checkwork:

