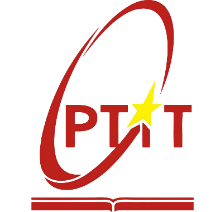
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA AN TOÀN THÔNG TIN**

-------🙞🙜🕮🙞🙜-------



**Báo cáo bài thực hành**

**TÌM HIỂU BÀI THỰC HÀNH**

**PTIT – PEN\_ATTK\_MITRE\_T1548**

Giảng viên: Nguyễn Ngọc Điệp

Môn học: Đồ án tốt nghiệp

Sinh viên: Đinh Việt Anh

Mã sinh viên: B20DCAT005

**Hà Nội - 2024**

Mục lục

[Bài thực hành: Tìm hiểu Setuid và Setgid (pen\_attk\_mitre\_t1548) 5](#_Toc186666106)

[1. Giới thiệu chung về bài thực hành: 5](#_Toc186666107)

[2. Nội dung và hướng dẫn bài thực hành 6](#_Toc186666108)

[3. Phân tích yêu cầu bài thực hành 7](#_Toc186666109)

[4. Thiết kế bài thực hành 7](#_Toc186666110)

[5. Cài đặt và cấu hình các máy ảo 9](#_Toc186666111)

[6. Tích hợp và triển khai 11](#_Toc186666112)

[7. Thử nghiệm và đánh giá 17](#_Toc186666113)

DANH MỤC CÁC HÌNH VẼ

[Hình 1. Giao diện Labedit của bài lab 11](#_Toc186666078)

[Hình 2. Cài đặt phần Result 11](#_Toc186666079)

[Hình 3. Dockerfiles của máy user 12](#_Toc186666080)

[Hình 4. Đăng nhập trong Labtainer 13](#_Toc186666081)

[Hình 5. Khởi tạo quá trình lưu trữ trên dockerhub 13](#_Toc186666082)

[Hình 6. Khai báo thông tin tài khoản 13](#_Toc186666083)

[Hình 8. Đăng nhập tài khoản dockerhub 14](#_Toc186666084)

[Hình 9. Quá trình tải bài lab lên dockerhub 15](#_Toc186666085)

[Hình 10. Lưu trữ bài lab trên dockerhub 16](#_Toc186666086)

[Hình 11. Lưu trữ bài lab trên github 17](#_Toc186666087)

[Hình 12. Tải bài lab về 17](#_Toc186666088)

[Hình 13. Chạy lab từ imodule 18](#_Toc186666089)

[Hình 14. Tìm kiếm chương trình chưa sticky bit 18](#_Toc186666090)

[Hình 15. Kiểm tra quyền thực thi của công cụ vừa tìm được 19](#_Toc186666091)

[Hình 16. Thực thi đã được thiết lâp sẵn trong folder root 19](#_Toc186666092)

[Hình 17. Đánh giá kết quả bài thực hành. 19](#_Toc186666093)

DANH MỤC BẢNG BIỂU

[Bảng 1. Bảng Result 9](#_Toc186666094)

[Bảng 2. Bảng Goal 10](#_Toc186666095)

DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| Từ viết tắt | Thuật ngữ tiếng Anh/Giải thích | Thuật ngữ tiếng Việt/Giải thích |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Bài thực hành: Tìm hiểu Setuid và Setgid (pen\_attk\_mitre\_t1548)

* + 1. Giới thiệu chung về bài thực hành:

Bài thực hành **"Tìm hiểu về Setuid và Setgid"** được thiết kế nhằm giúp sinh viên hiểu rõ cách kẻ tấn công có thể khai thác các chương trình được thiết lập bit **Setuid** và **Setgid** để thực hiện việc **nâng quyền** trên hệ thống Linux. Đây là một khía cạnh quan trọng trong bảo mật hệ thống, giúp sinh viên làm quen với các phương pháp khai thác lỗ hổng liên quan đến quản lý quyền trên hệ điều hành.

Trong bài thực hành này, sinh viên sẽ làm việc với **một container Attacker**, trong đó người dùng có quyền hạn thông thường và truy cập vào môi trường mục tiêu. Thông qua các bước thực hành, sinh viên sẽ học cách tìm kiếm, phân tích và khai thác các chương trình được cấu hình **Setuid** và **Setgid** để nâng quyền từ người dùng thường lên **root**.

Các kỹ năng mà sinh viên sẽ học được bao gồm:

* Tìm kiếm và nhận diện các chương trình thực thi được gắn bit **Setuid** và **Setgid** trên hệ thống Linux.
* Phân tích và xác định quyền truy cập của các chương trình này.
* Khám phá cách tận dụng các chương trình được cấu hình **Setuid/Setgid** để nâng quyền trên hệ thống.
* Hiểu rõ rủi ro bảo mật khi cấu hình sai các bit **Setuid** và **Setgid**.

Sau bài thực hành này, sinh viên sẽ nắm vững:

* Cách xác định các chương trình được thiết lập **Setuid** và **Setgid**.
* Cách khai thác các chương trình có cấu hình sai để **nâng quyền**.
* Hiểu rõ các rủi ro và biện pháp phòng chống việc lạm dụng **Setuid** và **Setgid** trong môi trường thực tế.

Thông qua bài thực hành, sinh viên sẽ nâng cao kỹ năng đánh giá bảo mật hệ thống Linux và có nhận thức rõ ràng hơn về các rủi ro tiềm ẩn khi quản lý quyền thực thi của các chương trình. Đây là bước đệm quan trọng giúp sinh viên phát triển kiến thức chuyên sâu về an toàn thông tin và các kỹ thuật phòng chống tương ứng trong thực tế.

* + 1. Nội dung và hướng dẫn bài thực hành

2.1 Mục đích

Bài thực hành giúp sinh viên tìm hiểu cách các chương trình được thiết lập **Setuid** và **Setgid** trên hệ điều hành Linux có thể bị khai thác để **nâng quyền** từ người dùng thường lên **root**. Thông qua việc phân tích và khai thác các chương trình thực thi này, sinh viên sẽ hiểu rõ hơn về các rủi ro bảo mật liên quan đến việc cấu hình sai quyền thực thi và tầm quan trọng của việc quản lý chặt chẽ các bit **Setuid** và **Setgid** trên hệ thống.

2.2 Yêu cầu đối với sinh viên

Sinh viên cần có kiến thức cơ bản về:

* **Hệ điều hành Linux** và các lệnh quản lý tập tin, quyền hạn người dùng.
* **Bit Setuid và Setgid**: hiểu ý nghĩa và cách chúng hoạt động.
* **Shell scripting** cơ bản để thực hiện các thao tác khai thác.
* **Công cụ phân quyền và thực thi** như find, ls, và bash để tìm kiếm và khai thác quyền hạn.

2.3 Nội dung thực hành

Khởi động bài lab:

Vào terminal, gõ:

*Labtainer -r <tên bài lab>*

*(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 1 terminal ảo sẽ xuất hiện, terminal này giả lập môi trường máy victim sau khi attacker đã truy cập thành công và tìm cách leo quyền từ user thường lên user root.

Trên terminal thực hiện tìm kiếm các chương trình thực thi mà được thiết lập bit setuid hoặc setgid:

*find / -perm u=s -type f 2>/dev/null*

Sau khi có kết quả tìm kiếm, dựa vào [GTFOBins](https://gtfobins.github.io/) kẻ tấn công kiểm tra các chương trình có thể thực hiện leo quyền và cách leo quyền

Sau khi kiểm tra kết quả trên, lệnh find được thiết lập bit suid có thể sử dụng để nâng quyền, kiểm tra bằng lệnh:

*find file\_name -exec whoami \;*

Nếu kết quả của lệnh trên là “root” => lệnh find được thiết lập bit suid với chủ sở hữu là root vì vậy chỉ cần được gọi đến là có thể thực thi dưới quyền root

Sử dụng lệnh find để thực thi script trong thư mục root

*find file\_name -exec python3 /root/script.py \;*

Kết thúc bài lab:

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

*stoplab <tên bài lab>*

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 <tên bài lab>*

1. Phân tích yêu cầu bài thực hành

Bài thực hành gồm 1 máy tính mô phỏng trường hợp attacker đã có quyền điều khiển máy victim dưới quyền người dùng bình thường, lợi dụng sticky bit được thiết lập không đúng cách để leo quyền lên người dùng quản trị. Để hoàn thành bài thực hành, sinh viên cần leo quyền lên người dùng root và thực thi script có sẵn trong thư mục root.

1. Thiết kế bài thực hành

Trên môi trường máy ảo Ubuntu được cung cấp, sử dụng docker tạo ra 1 container:

Cấu hình docker gồm có:

Tên máy: user

config: lưu cấu hình hoạt động của hệ thống

dockerfiles: mô tả cấu hình của container:

user: sử dụng các thư viện mặc định hệ thống và package .kali tích hợp sẵn script cần quyền root trong thư mục root và công cụ find được thiết lập sticky bit.

docs: lưu phần mô tả hướng dẫn làm bài thực hành cho sinh viên.

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

* ***Attacker*** tìm kiếm các chương trình thực thi được thiết lập suid hoặc sgid
* ***Attacker*** xác định được quyền truy cập của các chương trình tìm được
* ***Attacker*** tìm hiểu khả năng và cách nâng quyền của các chương trình
* ***Attacker*** biết được cách sử dụng chương trình và sử dụng nó thực thi script trong folder root quyền root.
* 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 Result

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Result Tag | Container | File | Field Type | Field ID | Timestamp Type | LINE ID |
| check\_user | user | .bash\_history | CONTAINS | whoami | File |  |
| Find\_sticky\_bit\_files | user | .bash\_history | CONTAINS | find / -perm -u=s | File |  |
| check\_user\_root | user | .bash\_history | CONTAINS | -exec whoami \; | File |  |
| run\_script | user | .bash\_history | CONTAINS | -exec python3 /root/script.py \; | File |  |
| output | user | secret | CONTAINS | Hash:\s([a-fA-F0-9]{64}) | File |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Goal ID | Goal Type | Operator | Result Tag | Answer Type | Parameter |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Bảng 2. Bảng Goal

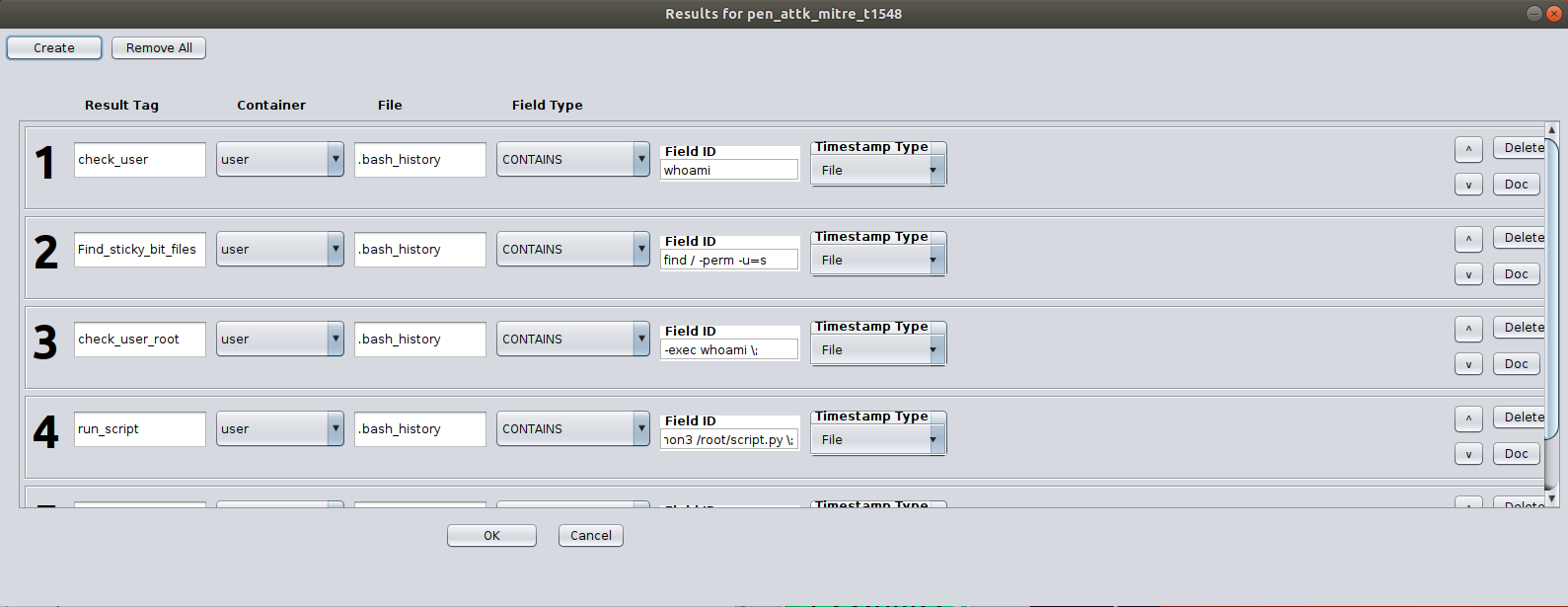
Bảng 3. Bảng Parameter

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Param ID | Operator | File name | Symbol | Step | Hashstring |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

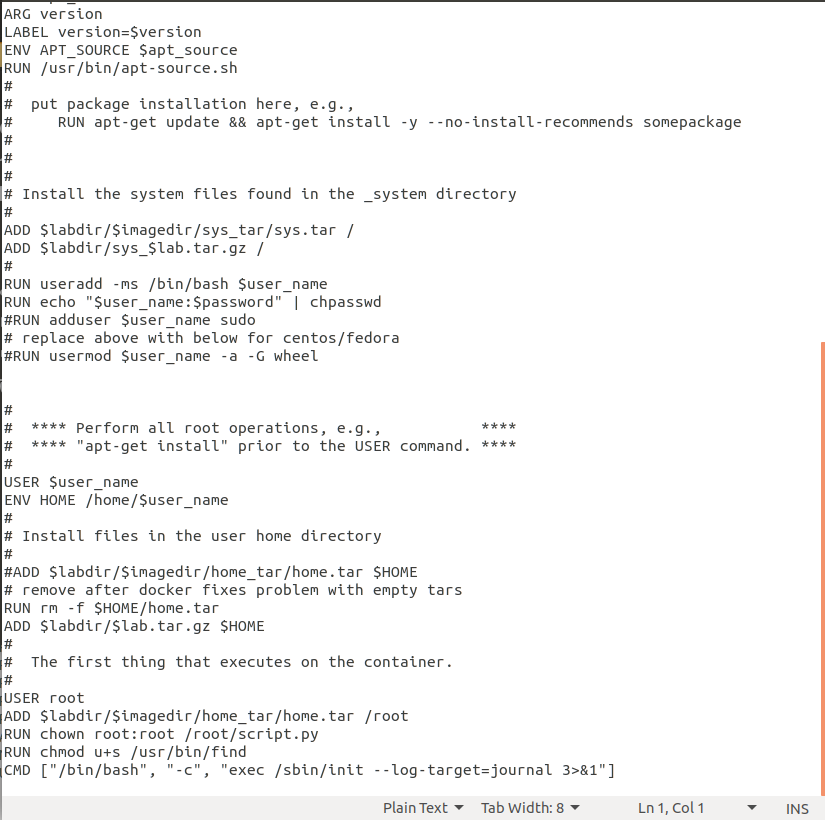
1. Cài đặt và cấu hình các máy ảo



Hình 1. Giao diện Labedit của bài lab



Hình 2. Cài đặt phần Result



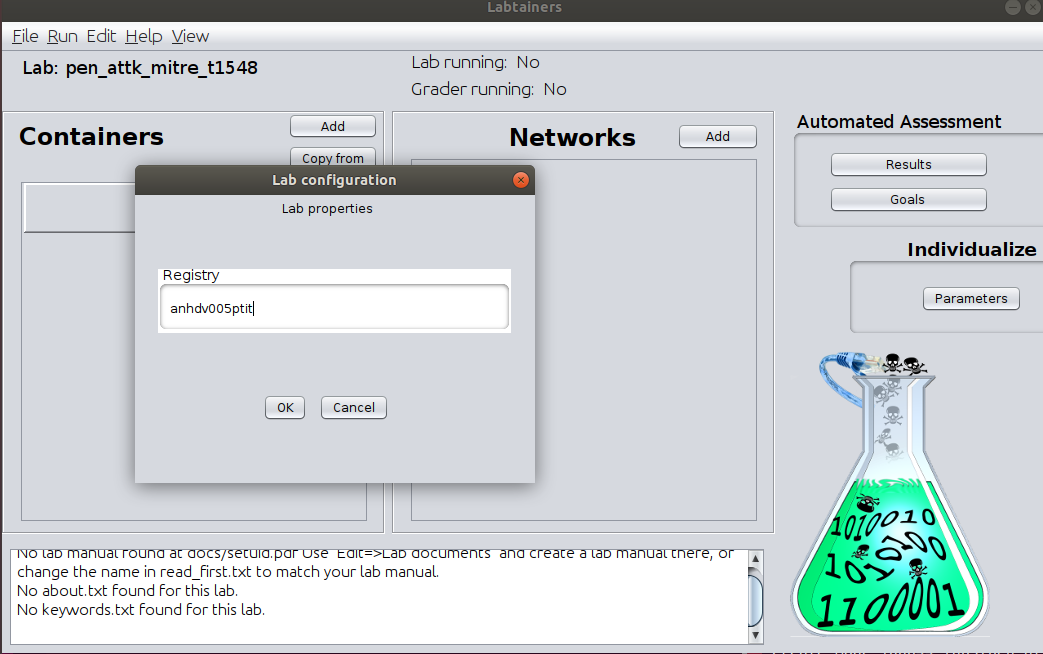
Hình 3. Dockerfiles của máy user

1. Tích hợp và triển khai

Bài thực hành đã được triển khai như sau:

* + - * 1. Docker Hub

Tạo tài khoản trên trang https://hub.docker.com trên giao diện labtainer chọn mục Edit/Config (registry) đănh nhập tên tài khoản hiển thị trên dockerhub và Build only lại.



Hình 4. Đăng nhập trong Labtainer

Khởi tạo quá trình lưu trữ trên dockerhub từ đường dẫn ~labtainer/trunk/labs với lệnh:

***git init (chỉ nhập 1 lần duy nhất)***

******

Hình 5. Khởi tạo quá trình lưu trữ trên dockerhub

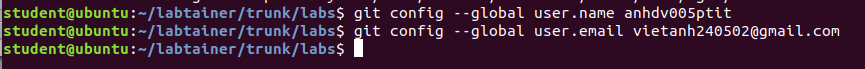
Khai báo các thông số để kết nối đến dockerhub với lệnh:

***git config --global user.name “your\_username”***

(với your\_username: tên hiển thị trên git)

***git config --global user.email “your\_email@example.com”***

(với your\_email@example.com: tài khoản đăng nhập dockerhub)

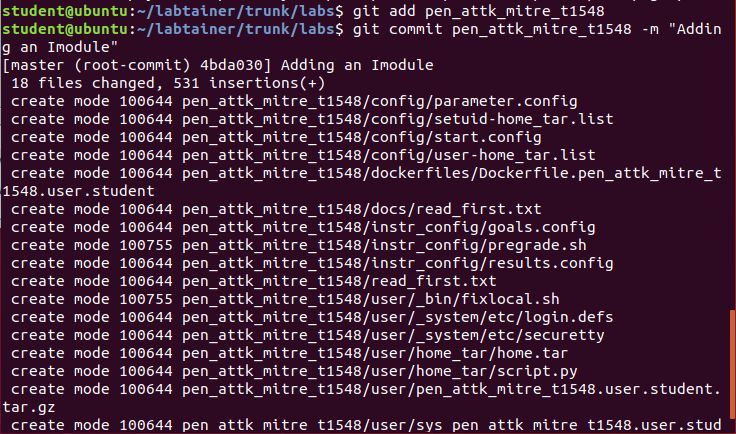


Hình 6. Khai báo thông tin tài khoản

Thêm bài lab muốn lưu trữ vào git theo lệnh:

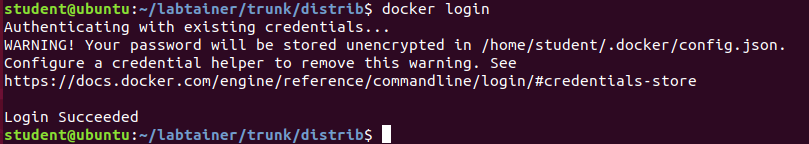
***git add <tên bài lab>***

***git commit <tên bài lab> -m “Adding an Imodule”***

******

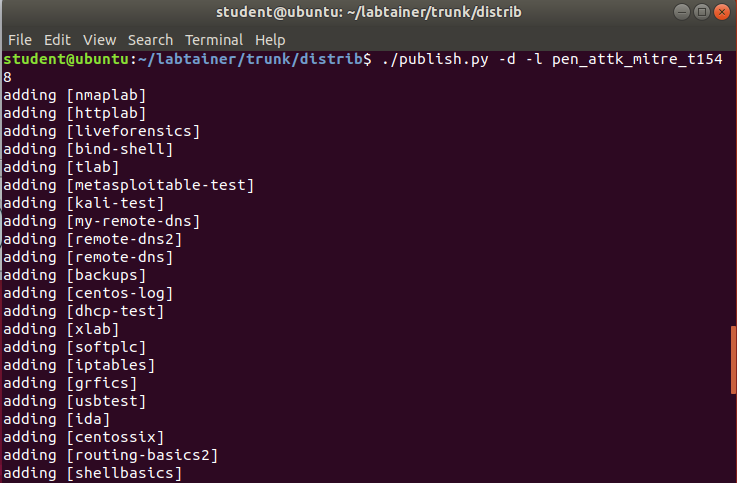
Hình 7. Thêm bài lab muốn lưu trữ

Chuyển đến thư mục ~trunk/distrib, sử dụng lệnh docker login để đăng nhập docker trên Labtainer



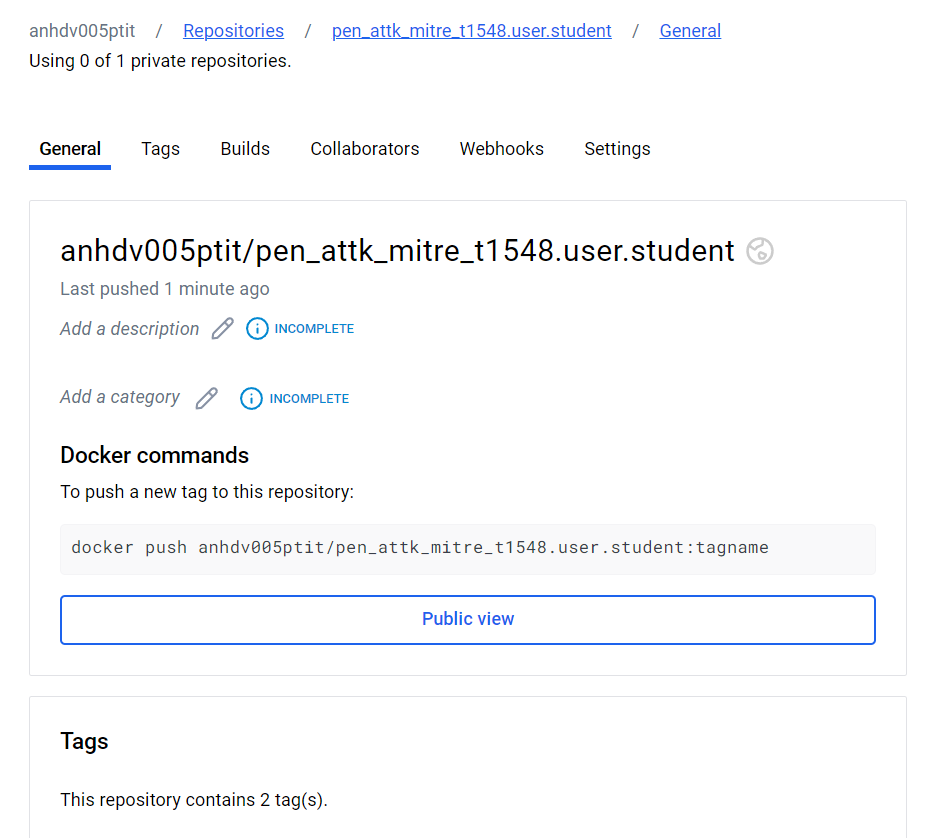
Hình 8. Đăng nhập tài khoản dockerhub

Đẩy bài lab lên dockerhub bằng cách sử dụng lệnh:



Hình 9. Quá trình tải bài lab lên dockerhub

Quá trình thành công các bài lab đã được đẩy lên quản lý trên docker hub



Hình 10. Lưu trữ bài lab trên dockerhub

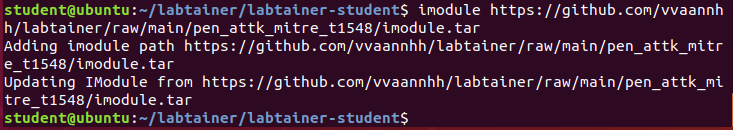
* + - * 1. Github



Hình 11. Lưu trữ bài lab trên github

Sinh viên tiến hành tải bài lab theo đường dẫn được cung cấp theo lệnh

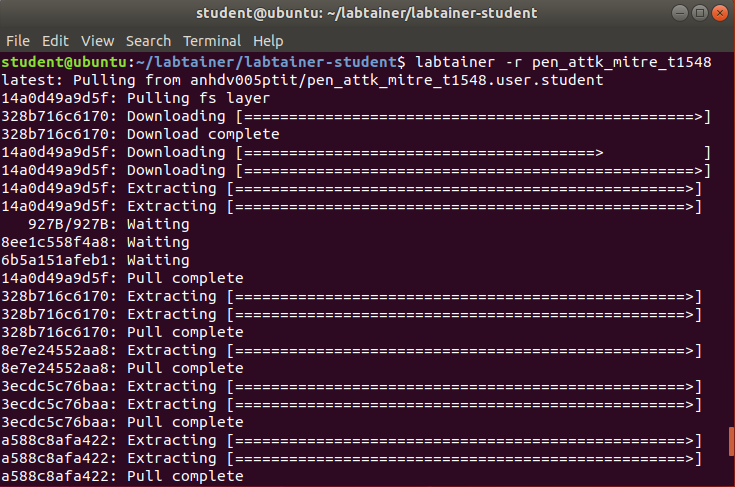
***imodule <đường dẫn>***



Hình 12. Tải bài lab về

Load bài lab từ imodule

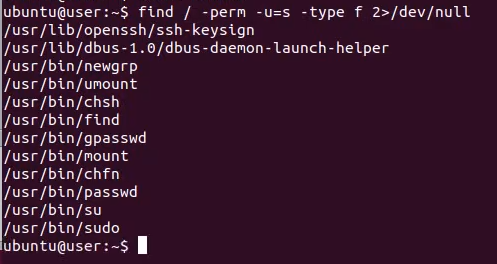
***labtainer -r <tên bài lab>***



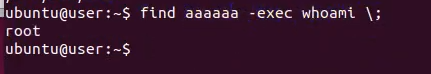
Hình 13. Chạy lab từ imodule

1. Thử nghiệm và đánh giá

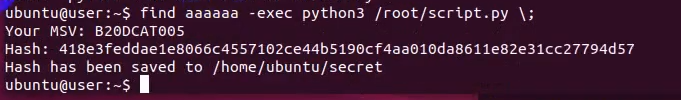
Bài thực hành đã được xây dựng thành công, dưới đây là hình ảnh minh họa về bài thực hành:



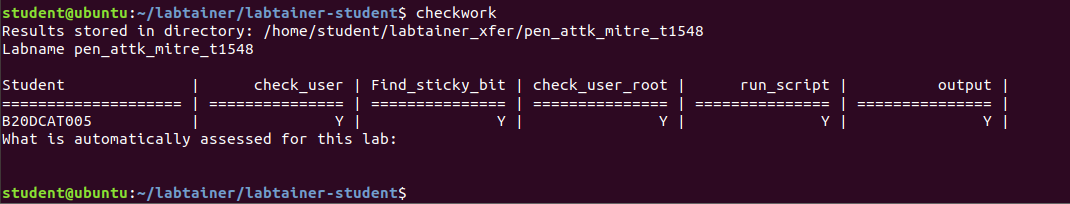
Hình 14. Tìm kiếm chương trình chưa sticky bit



Hình 15. Kiểm tra quyền thực thi của công cụ vừa tìm được



Hình 16. Thực thi đã được thiết lâp sẵn trong folder root



Hình 17. Đánh giá kết quả bài thực hành.