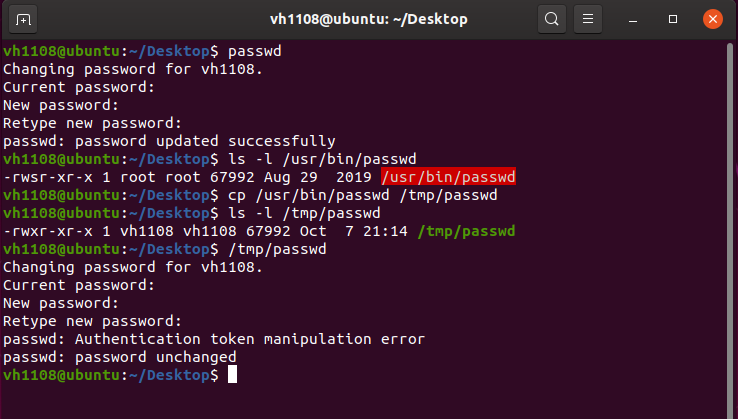
Lab 4

3.1

* - Các lệnh “passwd”, “chsh”, “su” và “sudo” cần chương trình Set-UID vì như vậy có thể cho phép user tạm có quyền thực thi file bằng chính quyền của chủ sở hữu. Chương trình Set-UID sẽ cấp quyền tạm thời cho user đang đăng nhập chạy file quyền của user tạo ra file đó (owner), user chạy sẽ có UID và cả GID của người tạo ra file đó.
* Nếu không có chương trình Set-UID các user sẽ không được cấp quyền của owner để chạy mội file thực thi không phải mình tạo ra hay command một lệnh nào đó ngoài quyền của user.

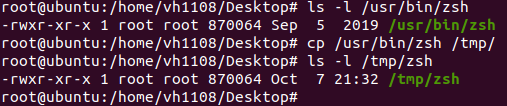


* Ban đầu có thể sử dụng lệnh “passwd” để thay đổi mật khẩu, kiểm tra thư mục /usr/bin/passwd thì thấy có Set-UID ( -rwsr-xr-x ).
* Copy thư mục /usr/bin/passwd vào /tmp/ và kiểm tra quyền thì không thấy set-UID, lúc này ta không thể thực thi /tmp/passwd

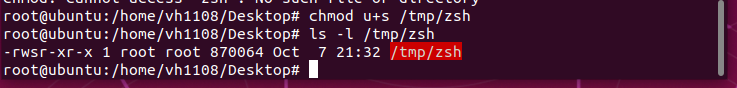
3.2

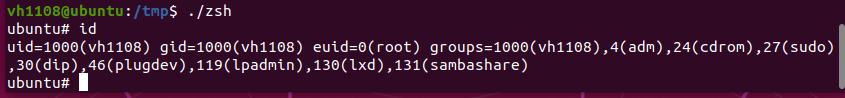
a/

* Đăng nhập vào root và copy /usr/bin/zsh vào /tmp/



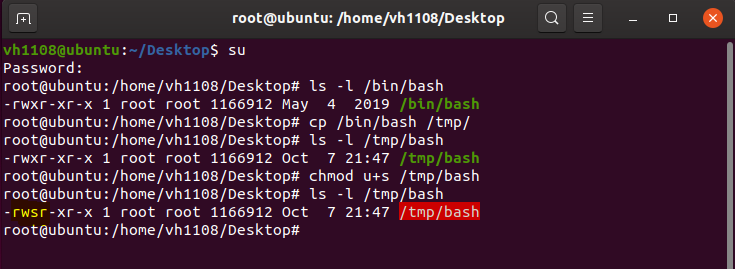
* Đặt /tmp/zsh thành chương trình set-root-UID với quyền 4755



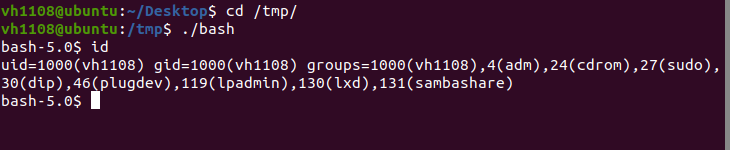
* Đăng nhập với một user bình thường và chạy /tmp/zsh
* 
* Lúc này, người dùng bình thường có được đặc quyền root

b/

* Đăng nhập vào root, sau đó sao chép /bin/bash vào /tmp/. Lúc này /tmp/bash chưa có set-UID, ta đặt chương trình set-UID cho nó và kiểm tra thì thấy đã có set-UID



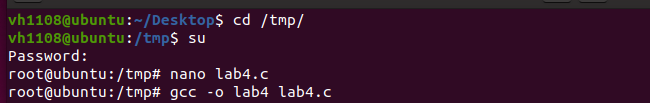
* Đăng nhập vào một người dùng bình thường và chạy /tmp/bash

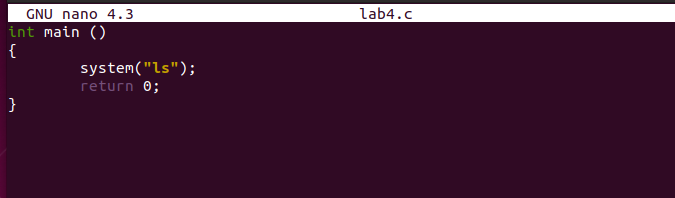


* Lúc này, người dùng bình thường KHÔNG có đặc quyền root. Ta làm tương tự như với /bin/zsh nhưng với thư mục bin/bash thì không được đặc quyền root.

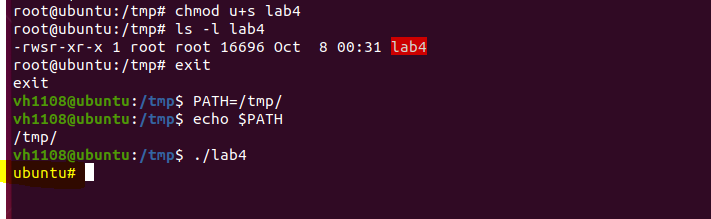
3.3

* Đăng nhập với quyền root, soạn file lab4.c với nội dung như hình và tiến hành biên dịch bằng gcc





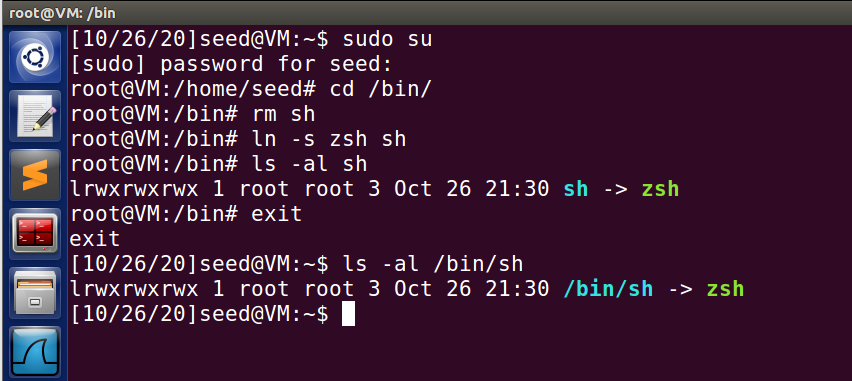
* Đặt set-UID cho file lab4 được dịch ở trên, kiểm tra lại quyền thì thấy đã thành công. Sau đó đăng nhập vào người dùng bình thường và cài đặt PATH ở /tmp/
* Cuối cùng ta tiến hành chạy file lab4 đã dịch ở trên



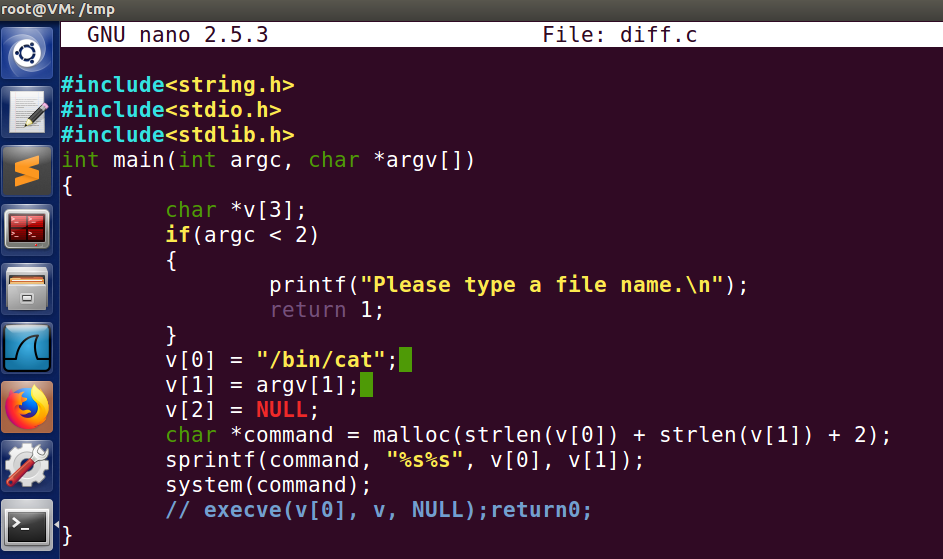
* Sau khi đã sửa đổi biếng môi trường, ta đã nhận được đặc quyền root

3.4

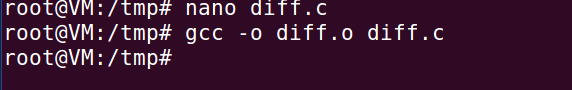
- Đăng nhập vào root, tạo liên kết giữa zsh và sh



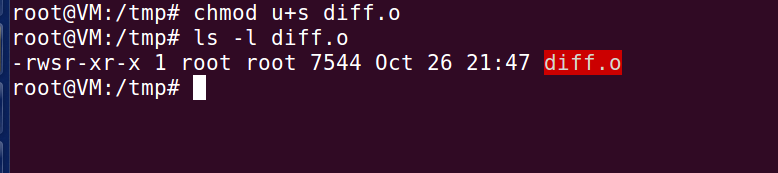
- Soạn file diff.c



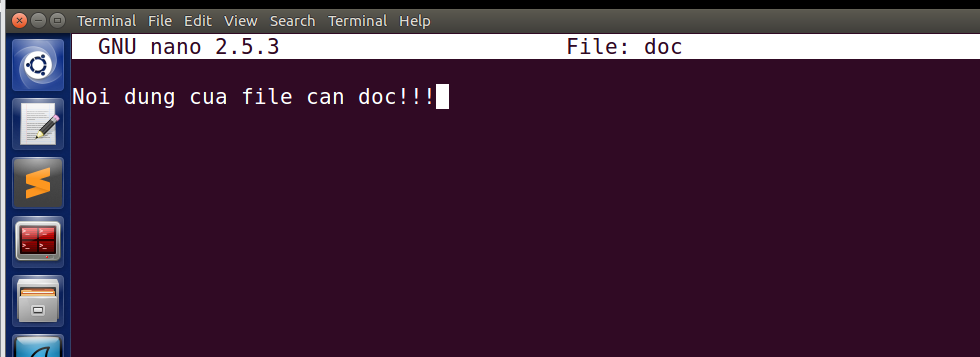
* Biên dịch file difference.c

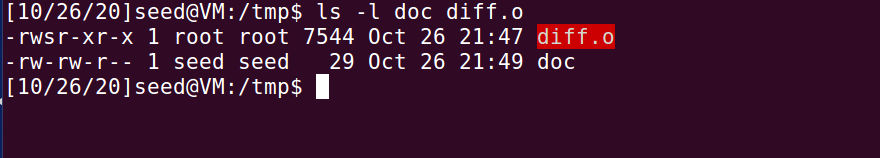


* Đặt set-uid

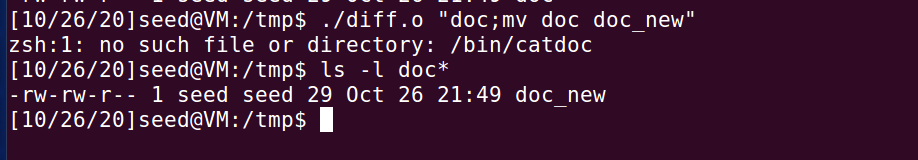


* Tạo một file doc để đọc với nội dung như sau

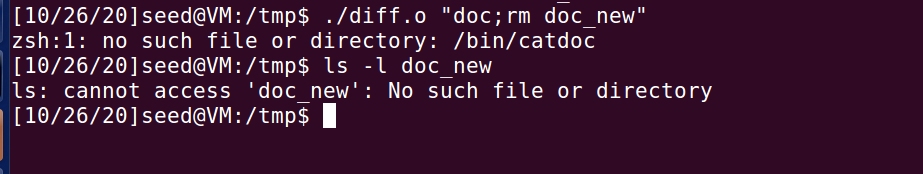




* Đổi tên file doc thành doc\_new



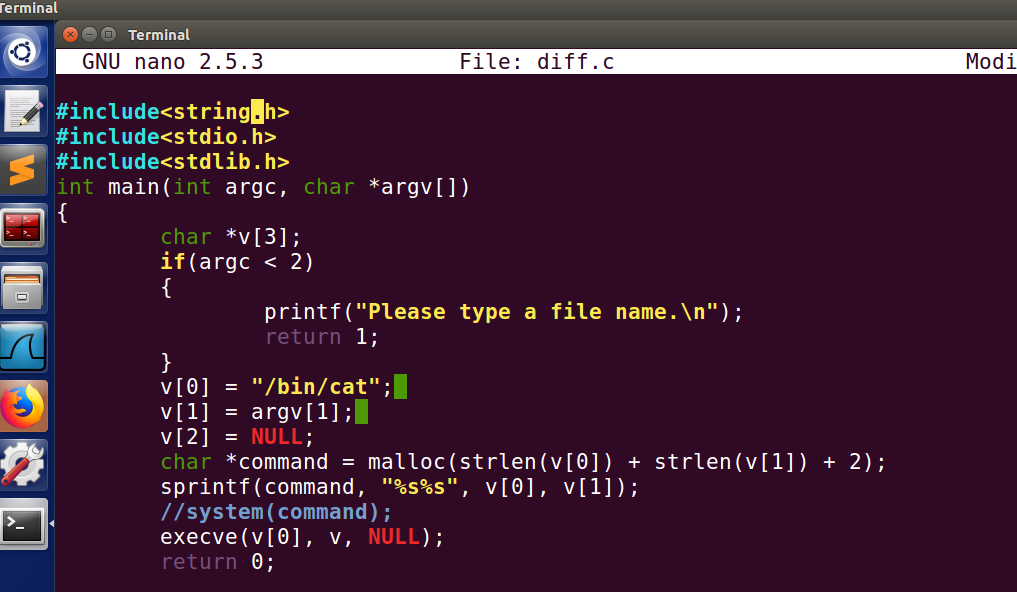
* Ví dụ xoá file doc\_new



* Các lệnh này không an toàn. Bob có thể tiến hành đọc, ghi và di chuyển các tập tin mà chỉ có người dùng root có quyền.

b/

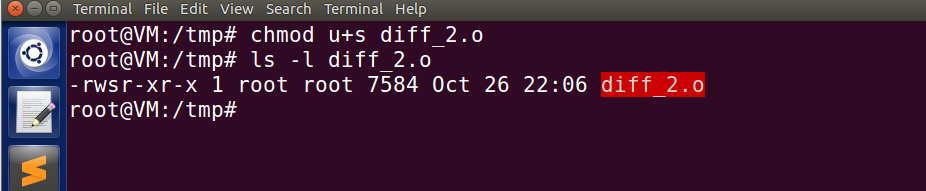
* Sử dụng execve() thay cho system()



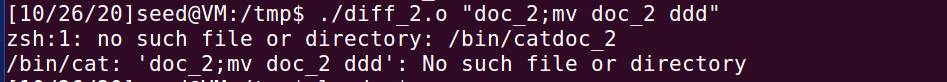
* Biên dịch file vừa sửa



* Đặt set-uid



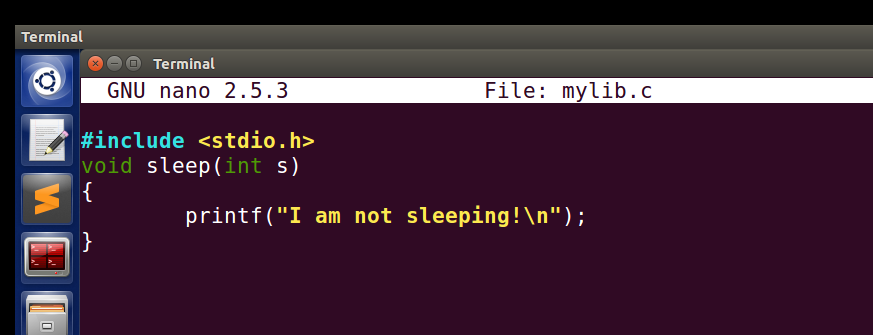
* Chạy file diff\_2.o



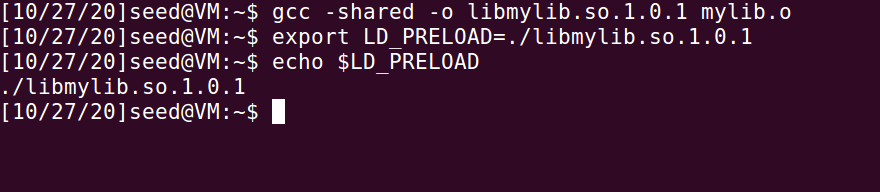
* Cuộc tấn công không thành công. Vì khi hệ thống gọi system() ( bin/sh liên kết với zsh). Sau khi chạy “cat file” với quyền root thì nó tiếp tục chạy “rm doc\_new”. Nhưng khi gọi execve() nó sẽ coi “doc\_2;rm doc\_2” như một tên file và tiến hành gọi và kết quả trả về là không có file tên “doc\_2;rm doc\_2”.

3.5

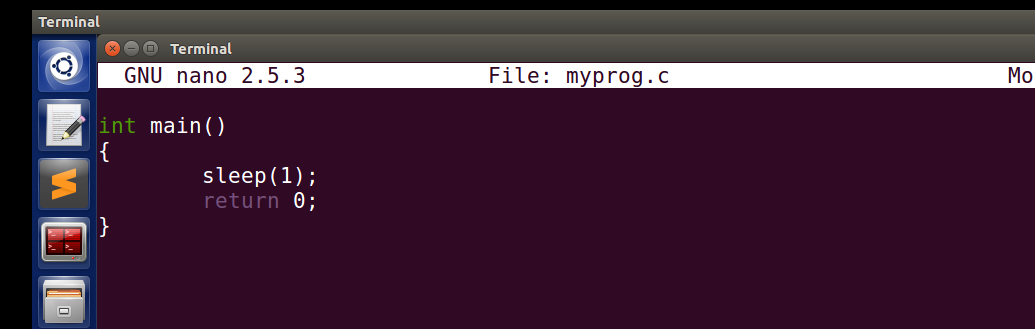
- Soạn file mylib.c



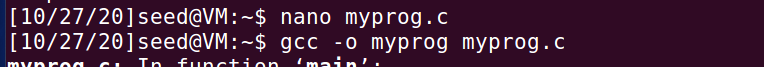
* Biên dịch file



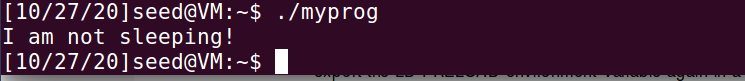
* Soạn file myprog.c



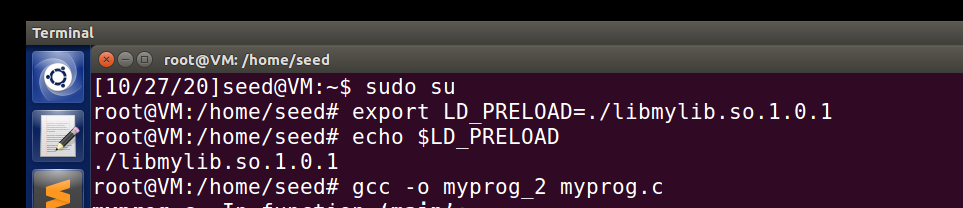
* Biên dịch file myprog.c

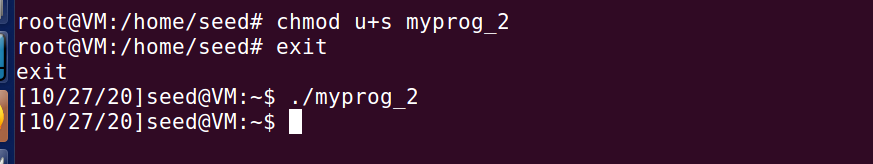


* Chạy file myprog với người dùng bình thường

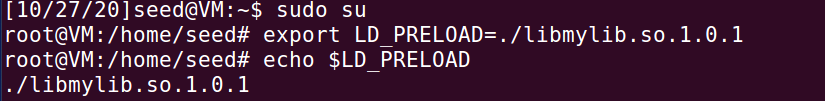


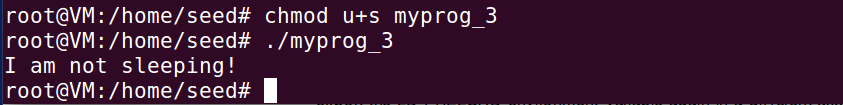
* Chạy file myprog.c với set-uid và người dùng bình thường





* Trong trường hợp này, nó sẽ bỏ qua biến môi trường LD\_PRELOAD và sử dụng hàm sleep () mặc định của hệ thống. Vì vậy, hàm sleep () sẽ không bị ghi đè.
* Chạy chương trình với set-uid và người dùng root





* Trong trường hợp này, nó sẽ sử dụng biến môi trường LD\_PRELOAD và ghi đè hàm sleep ().
* Tạo một user mới tên là user1, set-uid với user1, xuất lại biến môi trường và chạy chương trình với một người dùng không phải root