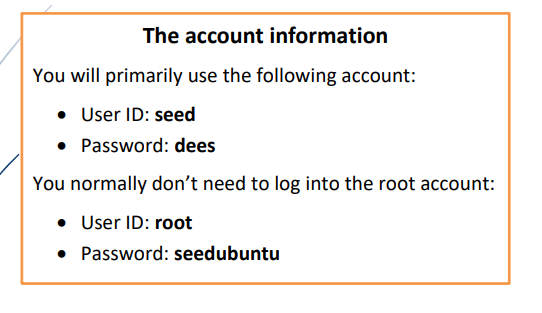
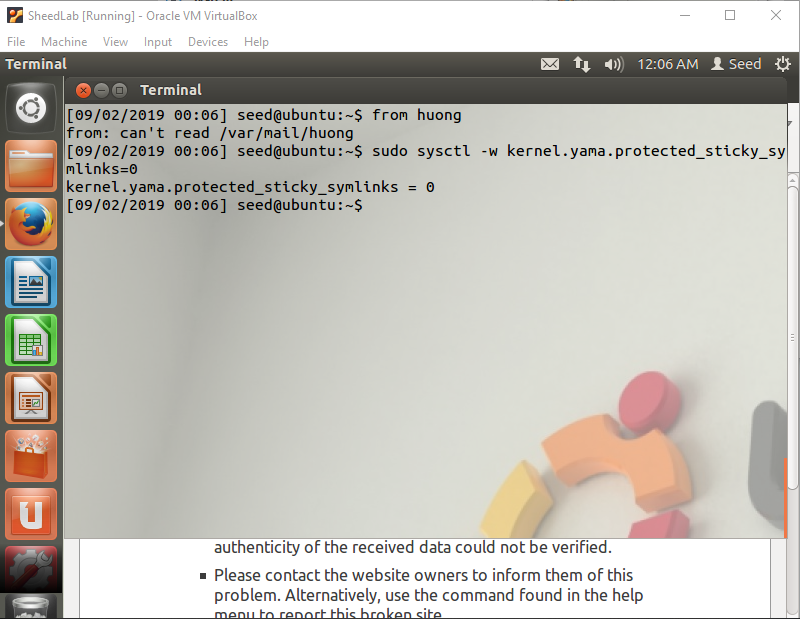
Task 1:

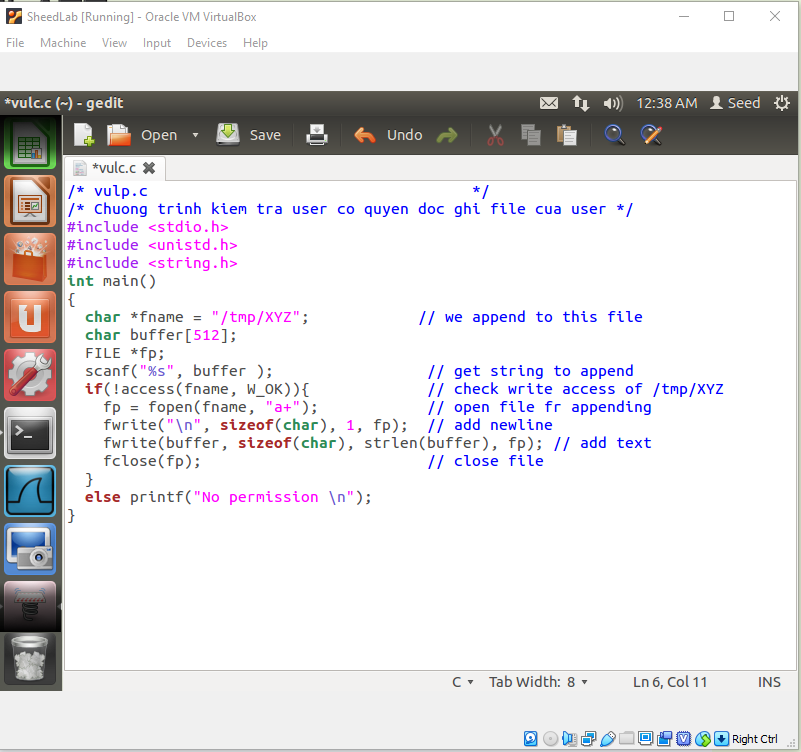
Đối với nhiệm vụ này, chúng ta sẽ vô hiệu hóa cơ chế bảo vệ - sticky protection mechanism for symbolic links – đây là một cơ chế bảo vệ cho các liên kết tượng trưng. Ở đây chúng ta sẽ cố gắng nối một dòng mới vào file passwd để chúng ta thể hiển điểm yếu race-condition. Chúng ta có một chương trình có điểm yếu là vulp.c và làm nó thành chương trình Set-UID được thiết lập bởi root. Sau đó chúng ta tạo một chương trình nhằm khai thác lỗ hổng race-condition bằng cách khai thác khoảng cách giữa thời gian kiểm tra và thời gian sử dụng(TOC/TOU) . Sau đó chúng ta tạo tệp input.txt chứa dữ liệu cần thêm vào tệp passwd. Test.txt là tệp thuộc sở hữu của người dùng “seed” và có đặc quyền của seed. Repeat.sh là một đoạn shell code lấy giá trị từ input.txt và liên tục đưa nó làm đầu vào cho chương trình vulp.c . Check.sh là một đoạn shell chạy để kiểm tra xem cuộc đua race-conditon có xảy ra hay không bằng cách so sánh tệp mật khẩu cũ với tệp mới.

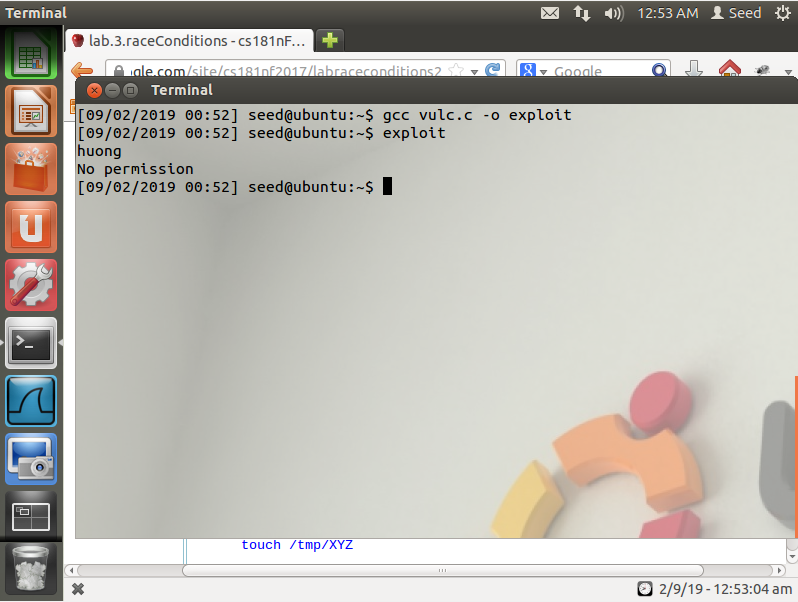


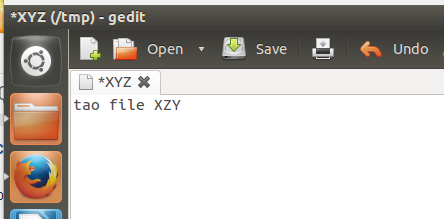
Tắt chết độ bảo vệ sticky symlinks :

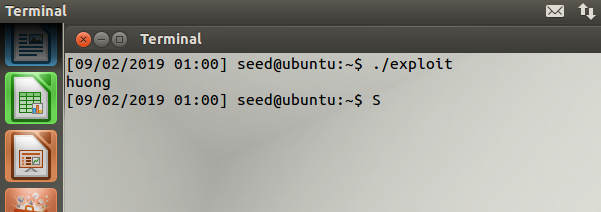


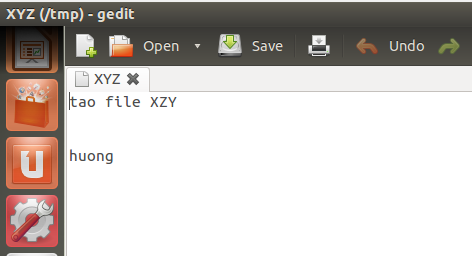
Đầu tiên ta cần chắc chắn chương trình Vulc.c hoạt động bình thường.





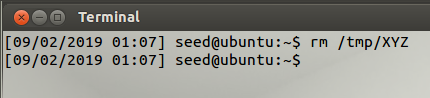
Sau khi chạy xong chương trình, ta nhập thử một chuỗi và kết quả trả về lúc này là No permission => người dùng Seed không có quyền ghi lên file /tmp/XYZ vì file đó chưa tồn tại. Sau khi tạo file /tmp/XYZ thì ta chạy lại chương trình sẽ có kết quả thành công.



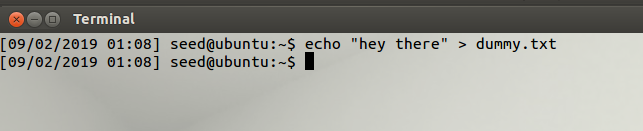


Lúc này file XYZ đã bị ghi lên vì người dùng có quyền ghi lên file.

Sau đó chúng ta cần link /tmp/XYZ đến file dummy.txt. Để file được link, ta cần phải chắc chắn rằng nó chưa link đến file nào khác, ta sẽ xóa file /tmp/XYZ bằng cách gõ lệnh :

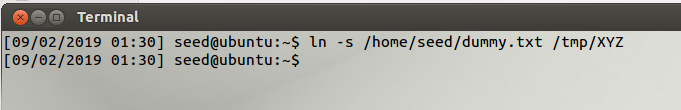


Tiếp theo sẽ tạo file dummy.txt

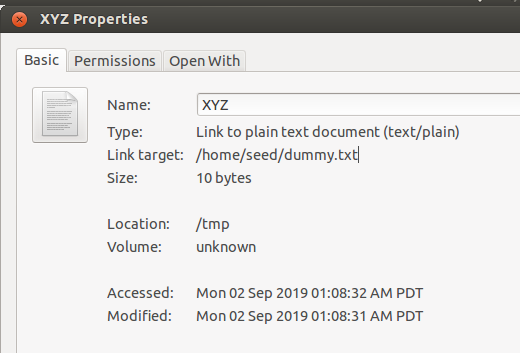




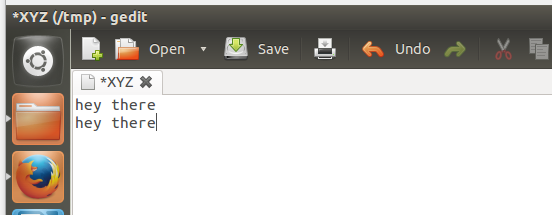
Cuối cùng ta sẽ tạo symbolic link từ /tmp/XYZ tới dummy.txt.



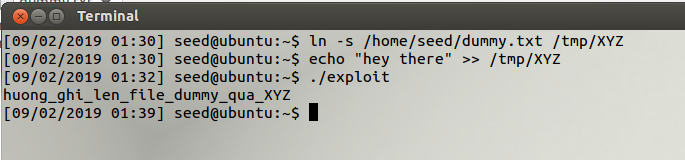
Lúc này file XYZ đã được link đến file dummy.txt

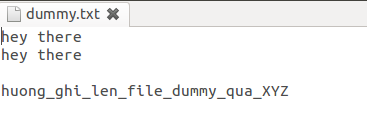


Để chắc chắn rằng tập tin đã được link chúng ta thử : >> - ghi đè



Sau đó chúng ta sẽ chạy chương trình vulc.c . Nhập input vào và kiểm tra xem file dummy.txt có được ghi chưa .

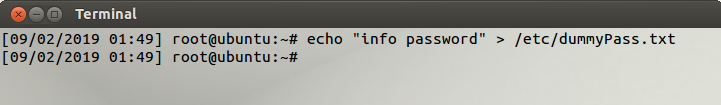


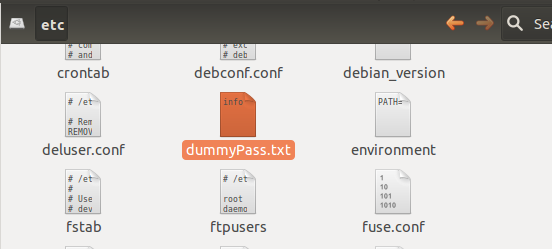


Tiếp theo ta sẽ tạo tệp hackerAccountInfo.txt và nối vào /etc/passwd . Mở /etc/passwd và xác định bản ghi cho root. Sau đó tạo một dòng text dưới người dùng – hacker . Đặt đoạn text vào trong file hackerAccountInfo.txt và chạy code.

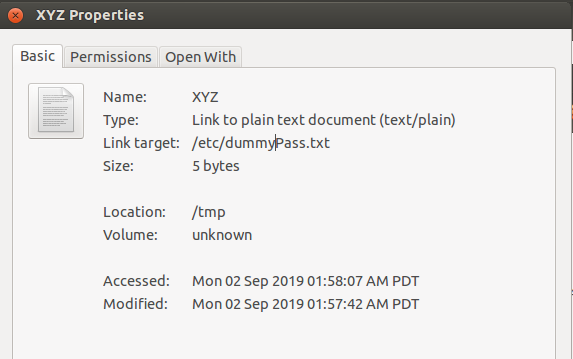
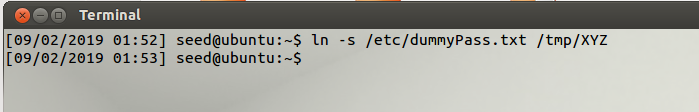
Kiểm tra để chắc chắn rằng thông tin tài khoản được thêm vào dummy.txt

Tao file dummyPass.txt trong thư mục /etc/ dưới quyền của root

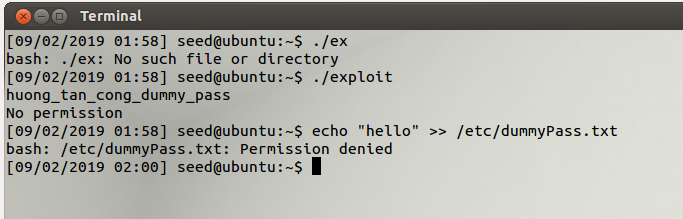




Link file dummyPass.txt của root với file /tmp/XYZ của user



Chạy chương trình để tấn công kết quả -> Không thể tấn công



Quan sát: Từ ảnh chụp màn hình ở trên, chúng ta có thể thấy nội dung của repeat.sh và check.sh.

Quan sát: Từ ảnh chụp màn hình ở trên, chúng ta có thể thấy nội dung đoạn mã exploit.c . Cuối cùng chúng tôi đã chạy tập lệnh shell repeat.sh trong một thiết bị đầu cuối khác và chạy khai thác.

Quan sát: Chúng ta có thể thấy sau nhiều lần khai thác race-condition, cuộc tấn công của chúng ta chạy và thông báo được hiển thị bởi check.sh.

Quan sát: Có thể thấy rằng tệp passwd của chúng tôi đã được thêm vào một người dùng mới với quyền root.

Giải thích : Trong nhiệm vụ này, chúng ta đang cố gắng sử dụng lỗ hổng race-condition và cố gắng sử dụng nó để khai thác khoảng thời gian giữa thời gian kiểm tra và thời gian sử dụng. /tmp và /var/tmp là các thư mục “world writable”- Thư mục mà bất cứ ai cũng có thể viết vào nhưng chỉ có người dùng mới có thể xóa hoặc di chuyển các tệp của mình trong thư mục này vì tính năng bảo vệ - sticky symlinks protection. Vì vậy trong lần thực hành này chúng ta sẽ tắt tính năng bảo vệ sticky symlinks protection để người dùng có thể theo dõi liên kết tương trưng của các thư mục world writable. Nếu bảo vệ nàyđược bật thì chúng ta không thể theo dõi liên kết tương trưng của người dùng khác như thư mục /tmp . Chúng ta sẽ cố gắng tạo điểm liên kết tương trung cho tệp do người dùng sở hữu ban đầu và sau đó huỷe liên kết tệp đó và chỉ đến tệp sở hữu của root để có thể khai thác và thay đổi tệp của root. Để bảo vệ điều này, các chương trình Set-UID kiểm tra các file thay đổi đã được thiếp lập, chương trình sử dụng access() để kiểm tra UID hợp lệ và kiểm tra fopen() cho UID hợp lệ. Vì vậy mục tiêu chúng ta là trỏ đến tệp do người dùng sở hữu để vượt qua access() và trỏ đến tệp sở hữu của root như tệp password trước khi mở file fopen() vì đây là chương trình Set-UID và EUID là root. Điều này sẽ vượt qua được kiểm tra fopen() và chúng ta có quyền truy cập vào tệp mật khẩu. Sau đó chúng ta thêm ngườid ùng mớivào hệ thống. Với nhiều lần thử từ người dùng, chúng ta có thể khai thác cửa sổ này.

Task 2 :

Quan sát: Trong tác vụ này, chúng tôi thay đổi chương trình dễ bị tổn thương thành mã như được hiển thị và thực hiện cùng một nhiệm vụ như trước. Chúng tôi thấy rằng cuộc tấn công đã không thành công ngay cả sau 3 giờ.