Теоретичні відомості

У UNIX/Linux-файлових системах, щоб запустити файл як програму або скрипт, необхідно, щоб були виставлені права на виконання (x). Однак існує виняток, якщо скрипт не має прав на виконання, але має права на читання, його можна передати як аргумент до інтерпретатора напряму, наприклад:

bash script.sh

Тут відбувається не запуск скрипта безпосередньо, а виклик інтерпретатора (bash), який читає вміст файлу та виконує його. Таким чином, обхід прав на виконання — можливий.

Приклад

Чи може користувач запускати скрипт, не маючи на нього прав виконання? Обґрунтуйте і протестуйте

Створимо файл myscript.sh:

echo 'echo "Hello from script!"' > myscript.sh

chmod -x myscript.sh # Видаляємо права на виконання

bash myscript.sh

Результат:

Hello from script!

звичайний запуск:

./myscript.sh

Результат:

bash: ./myscript.sh: Permission denied

Права на виконання потрібні лише для прямого запуску скрипта (./script.sh). Але якщо ми вказуємо інтерпретатор (bash script.sh, python script.py), потрібні лише права на читання.

Це часто використовується:

- у автоматизації,
- в тестуванні безпеки (обхід обмежень),
- при аналізі прав доступу.

Загальні завдання

Завдання 9.1

Напишіть програму, яка читає файл `/etc/passwd` за допомогою команди `getent passwd`, щоб дізнатись, які облікові записи визначені на вашому комп'ютері.

Програма повинна визначити, чи є серед них звичайні користувачі (ідентифікатори UID повинні бути більші за 500 або 1000, залежно від вашого дистрибутива), окрім вас.

Завдання 9.2

Напишіть програму, яка виконує команду `cat /etc/shadow` від імені адміністратора, хоча запускається від звичайного користувача.

(Ваша програма повинна робити необхідне, виходячи з того, що конфігурація системи дозволяє отримувати адміністративний доступ за допомогою відповідної команди.)

Завдання 9.3

Напишіть програму, яка від імені `root` копіює файл, який вона перед цим створила від імені звичайного користувача. Потім вона повинна помістити копію у домашній каталог звичайного користувача.

Далі, використовуючи звичайний обліковий запис, програма намагається змінити файл і зберегти зміни. Що відбудеться?

Після цього програма намагається видалити цей файл за допомогою команди `rm`. Що відбудеться?

Завдання 9.4

Напишіть програму, яка по черзі виконує команди `whoami` та `id`, щоб перевірити стан

облікового запису користувача, від імені якого вона запущена.

Є ймовірність, що команда `id` виведе список різних груп, до яких ви належите. Програма повинна це продемонструвати.

Завдання 9.5

Напишіть програму, яка створює тимчасовий файл від імені звичайного користувача. Потім від імені суперкористувача використовує команди `chown` і `chmod`, щоб змінити тип володіння та права доступу.

Програма повинна визначити, в яких випадках вона може виконувати читання та запис файлу, використовуючи свій обліковий запис.

Завдання 9.6

Напишіть програму, яка виконує команду `ls -l`, щоб переглянути власника і права доступу до файлів у своєму домашньому каталозі, в `/usr/bin` та в `/etc`.

Продемонструйте, як ваша програма намагається обійти різні власники та права доступу користувачів, а також здійснює спроби читання, запису та виконання цих файлів.

Варіанти завдань

1. Дослідіть ситуацію, коли користувач може змінити файл, хоча не є його власником. Наведіть можливі сценарії та протестуйте їх.

2. Створіть систему, де обмежується доступ до певного файлу лише в певні години. Які інструменти можна використати?

3. Визначте, чи можна прочитати вміст файлу, якщо всі дозволи на нього зняті. Поясніть, у яких випадках це можливо.

4. Створіть файл із нестандартними правами доступу та перевірте його поведінку в різних файлових системах (ext4, tmpfs, NFS).

5. Змоделюйте ситуацію, де користувач має лише права на запис до файлу, але не може його прочитати. Як це можливо, і які наслідки для безпеки або роботи програм?

6. Напишіть програму, яка визначає, чи є користувач "системним" на основі нестандартних критеріїв (не UID).

7. Змоделюйте ситуацію, коли `getent passwd` повертає неповні або некоректні дані. Чому це може статись?

8. Створіть групу користувачів, яка має повний доступ до окремої директорії, але жоден з учасників не має доступу до неї окремо.

9. Чи можна мати два різні облікові записи з однаковим UID? Які наслідки?

10. Визначте, які приховані механізми можуть дати доступ до закритого ресурсу без зміни прав доступу.

11. Чим відрізняється виконання `id` у shell та з системного виклику? Протестуйте в різних середовищах.

12. Дослідіть поведінку `fork()` у випадку, коли системні ресурси обмежено. Як змінюється вихід?

13. Створіть "обманну" програму, яка виглядає як `ls`, але виконує інші дії. Як її виявити?

14. Змоделюйте, як `cron` виконує команду з правами іншого користувача. Коли це не спрацює?

15. Напишіть скрипт, що веде логування команд користувача. Як приховати його від `ps`?

16. Виконайте експеримент: що відбувається при зміні прав на `/etc/shadow` через помилку в скрипті.

17. Які загрози можуть виникнути, якщо у користувача є доступ до `sudo`, але без пароля?

18. Побудуйте сценарій, де користувач має доступ до конфіденційної інформації через тимчасові файли.

19. Чи можна виконати команду як root без використання `sudo` або `su`? Шукайте нестандартні підходи.

20. Спробуйте виконати команду як інший користувач, не змінюючи UID.

21. Як змінюється доступ до файлу, якщо його жорстке посилання видалено, але процес ще використовує файл?

22. Змоделюйте ситуацію, коли файл є, але система каже, що він відсутній.

23. Виконайте серію дій з `chown`, щоб об'єкт мав змішані права доступу у різних процесів.

24. Дослідіть, як права доступу до каталогу впливають на доступ до файлів у ньому.

25. Як змінюється доступність файлу, якщо mount-пункт перекриває його розташування?

26. Створіть систему, де користувач запускає програму, яка сама змінює права на інші файли. Які ризики?

27. Створіть набір тестів, які автоматично визначають зміну доступності системних ресурсів залежно від дій користувача.