Title: Security Analysis of File Permissions and User Access in Linux Operating Systems

1. Experiment aim:

This comprehensive experiment is designed to immerse students in the intricate realm of file permissions and user access control within a Linux operating system. Through a series of hands-on exercises, participants will delve into the nuanced aspects of the chmod command, explore the dynamic interplay between user groups and file security, set up file access logging and monitoring systems, and navigate advanced security measures. The primary objective is to furnish participants with practical insights into Linux file permission systems, fostering a profound understanding of the principles governing user access and system security.

By engaging in these exercises, students will not only execute essential commands on the Linux terminal but also gain a holistic understanding of how their actions reverberate through the system. Through trial and exploration, participants will discern the implications of altering permissions, the significance of user groups in access control, the necessity of file access logging for security audits, and the potential advantages offered by advanced security measures.

The experiment aims to empower participants to think critically about security implications and strategic decision-making when configuring file permissions. It endeavors to bridge the gap between theoretical knowledge and practical application, enabling students to develop proficiency in securing file access in Linux environments. Upon completion, participants will be equipped with a toolkit of skills that extends beyond textbook concepts, allowing them to implement robust security measures in real-world scenarios, reinforcing the practical significance of their learning.

   
2. Theoretical background:

**Linux File Permissions:** Linux systems implement a robust file permission system to regulate access to files and directories. Permissions are categorized into three levels: user (owner), group, and others. Each level can be granted read, write, and execute permissions independently.

**chmod Command:** The chmod command is a fundamental tool for altering file permissions in a Linux environment. It allows users to modify the read, write, and execute permissions for files and directories. The command utilizes a numerical representation or symbolic notation to define the permission settings.

User Access Control: Understanding how different users and groups interact with files is crucial for maintaining a secure system. Users are classified into three categories: owner, group, and others. Proper management of user permissions ensures that only authorized individuals or processes can access, modify, or execute files.

**User Groups and File Security:** User groups provide an additional layer of access control. By assigning users to specific groups, administrators can streamline permissions management. Group ownership of files allows multiple users to share access, facilitating collaboration while maintaining security.

**File Access Logging and Monitoring:** Logging and monitoring file access activities are integral components of a robust security strategy. Tools like auditd or system logs enable administrators to track user interactions with files, aiding in the detection of unauthorized access or suspicious activities.

**Advanced Security Measures:** Beyond basic file permissions, Linux offers advanced security measures such as Access Control Lists (ACLs), AppArmor, and SELinux. ACLs allow for more granular control over file access, while security frameworks like AppArmor and SELinux provide additional layers of protection by confining the actions of processes.

By comprehending these theoretical concepts, participants will be well-prepared to navigate the subsequent practical exercises. The theoretical knowledge serves as the foundation for understanding the significance of different file permission configurations, user access controls, and advanced security measures in Linux operating systems.

3. Research:

**Ex. 1. Exploring chmod Commands Scenario:**

1. Create a new directory named "SecureFiles" in your home directory.

2. Inside "SecureFiles," create three text files: "file1.txt," "file2.txt," and "file3.txt."

3. Set the following permissions:

* "file1.txt" should be readable, writable, and executable by the owner, and readable by others.
* "file2.txt" should be readable and writable by the owner only.
* "file3.txt" should be readable, writable, and executable by the owner and the group.

# Scenario: Create a new directory named "SecureFiles" in your home directory.

mkdir ~/SecureFiles

# Scenario: Inside "SecureFiles," create three text files: "file1.txt," "file2.txt," and "file3.txt."

touch ~/SecureFiles/file1.txt ~/SecureFiles/file2.txt ~/SecureFiles/file3.txt

# Scenario: Set permissions for "file1.txt" - readable, writable, and executable by the owner, and readable by others.

chmod 744 ~/SecureFiles/file1.txt

# Scenario: Set permissions for "file2.txt" - readable and writable by the owner only.

chmod 600 ~/SecureFiles/file2.txt

# Scenario: Set permissions for "file3.txt" - readable, writable, and executable by the owner and the group.

chmod 770 ~/SecureFiles/file3.txt

**Questions:**

What chmod commands did you use to set the specified permissions?

For "file1.txt": chmod 744 ~/SecureFiles/file1.txt

For "file2.txt": chmod 600 ~/SecureFiles/file2.txt

For "file3.txt": chmod 770 ~/SecureFiles/file3.txt

How do the permissions of each file affect user access?

For "file1.txt": The owner has read, write, and execute permissions, while others have only read permissions.

For "file2.txt": The owner has read and write permissions, while others have no permissions.

For "file3.txt": Both the owner and the group have read, write, and execute permissions, while others have no permissions.

**Ex. 2.  User Groups and File Access Scenario:**

1. Create a new user named "TestUser" on your Linux system.
2. Add "TestUser" to a group named "SecureGroup."
3. Ensure that "SecureGroup" has read and write access to all files inside the "SecureFiles" directory.
4. Log in as "TestUser" and attempt to modify "file3.txt" inside the "SecureFiles" directory.

# Scenario: Create a new user named "TestUser" on your Linux system.

sudo adduser TestUser

# Scenario: Add "TestUser" to a group named "SecureGroup."

sudo addgroup SecureGroup

sudo usermod -aG SecureGroup TestUser

# Scenario: Ensure that "SecureGroup" has read and write access to all files inside the "SecureFiles" directory.

chmod -R g+rw ~/SecureFiles

# Scenario: Log in as "TestUser" and attempt to modify "file3.txt" inside the "SecureFiles" directory.

su - TestUser

echo "Additional content" >> ~/SecureFiles/file3.txt

exit

**Questions:**

How did you add "TestUser" to the "SecureGroup"?

Used the command sudo usermod -aG SecureGroup TestUser to add the user "TestUser" to the group "SecureGroup."

Why was "TestUser" unable to modify "file3.txt"? What permissions were missing?

"TestUser" was unable to modify "file3.txt" because the write permissions for the group were not granted. The command chmod -R g+rw ~/SecureFiles was used to ensure read and write access for the group "SecureGroup" to all files inside the "SecureFiles" directory.

**Ex. 3. Logging and Monitoring File Access Scenario:**

1. Enable file access logging for the "SecureFiles" directory.
2. Create a log file to record all file access events.
3. Access "file1.txt" from another user account, and check the log file for the recorded event.

# Scenario: Enable file access logging for the "SecureFiles" directory.

sudo auditctl -w ~/SecureFiles -p rwxa

# Scenario: Create a log file to record all file access events.

sudo touch /var/log/file\_access.log

sudo chmod 666 /var/log/file\_access.log

# Scenario: Access "file1.txt" from another user account and check the log file for the recorded event.

# (Assuming another user account is named "AnotherUser")

su - AnotherUser

cat ~/SecureFiles/file1.txt

exit

cat /var/log/file\_access.log

**Questions:**

How did you enable file access logging for the directory?

Used the command sudo auditctl -w ~/SecureFiles -p rwxa to enable file access logging for the "SecureFiles" directory.

What information is logged when accessing "file1.txt" from another user account?

The log file /var/log/file\_access.log will record events related to accessing "file1.txt," capturing details such as the username, timestamp, and the type of access (read, write, execute). The exact information can be extracted from the log file for further analysis.

4. Conclusions:

This series of practical exercises has provided a comprehensive exploration of file permissions, user access control, and file access logging within a Linux operating system. By delving into the intricacies of chmod commands, user groups, and access monitoring, participants gained valuable insights into securing files and directories.

File Permissions and chmod Commands: Participants successfully utilized chmod commands to set precise permissions for individual files within the "SecureFiles" directory. The commands demonstrated the flexibility of Linux file permission systems, allowing users to tailor access rights at both the user and group levels. These exercises highlighted the importance of understanding the numerical representations and symbolic notations used by chmod for effective permission management.

User Groups and File Security: The introduction of a new user ("TestUser") and its addition to the "SecureGroup" showcased the role of user groups in file access control. By granting read and write access to the group for all files inside the "SecureFiles" directory, participants enhanced their understanding of how groups contribute to collaborative workspaces while maintaining security. The inability of "TestUser" to modify "file3.txt" emphasized the significance of group permissions in determining user access.

File Access Logging and Monitoring: Enabling file access logging for the "SecureFiles" directory provided participants with a glimpse into the world of access monitoring. The log file recorded events associated with file access, offering a comprehensive view of user activities. By accessing "file1.txt" from another user account, participants observed firsthand how logging mechanisms capture essential information such as usernames, timestamps, and access types. This exercise illustrated the pivotal role of file access logging in enhancing system security through continuous monitoring and auditing.

In conclusion, these exercises have equipped participants with practical skills and knowledge essential for effective file access management in Linux environments. Understanding and manipulating file permissions, leveraging user groups, and implementing access logging mechanisms contribute to creating secure and well-organized file systems. As students apply these principles in real-world scenarios, they will be better prepared to navigate the complexities of access control and security within Linux operating systems.