Author: William Brown

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

1. Experiment aim:

**Student’s answer:**

The core aim of this experiment is to investigate the practical implications of configuring file access logging and monitoring mechanisms within a Linux system. By enabling file access logging, creating log files, and analyzing access events, the goal is to evaluate the effectiveness of these monitoring techniques in enhancing system security.  
   
2. Theoretical background:

**Student’s answer:**

The experiment is grounded in the concept of file access logging as a critical security measure in monitoring and tracking file interactions within a Linux environment. By recording access events, including user details and timestamps, file access logging provides valuable insights into system activities and potential security breaches.  
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.

**Student’s answer:**

# Create a new directory named "SecureFiles" in the home directory

mkdir ~/SecureFiles

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

touch ~/SecureFiles/file1.txt

touch ~/SecureFiles/file2.txt

touch ~/SecureFiles/file3.txt

# Set permissions for "file1.txt"

chmod 744 ~/SecureFiles/file1.txt

# Set permissions for "file2.txt"

chmod 600 ~/SecureFiles/file2.txt

# Set permissions for "file3.txt"

chmod 770 ~/SecureFiles/file3.txt

**Questions:**

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

**Student’s answer:**

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

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

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

How do the permissions of each file affect user access?

**Student’s answer:**

"file1.txt":

* Owner: Read, Write, Execute
* Group: Read
* Others: Read

"file2.txt":

* Owner: Read, Write
* Group: No access
* Others: No access

"file3.txt":

* Owner: Read, Write, Execute
* Group: Read, Write, Execute
* Others: No access

**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.

**Student’s answer:**

# Add a new user named "TestUser" to the Linux system

sudo adduser TestUser

# Add "TestUser" to a group named "SecureGroup"

sudo addgroup SecureGroup

sudo usermod -aG SecureGroup TestUser

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

chmod -R 660 ~/SecureFiles

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

su - TestUser

nano ~/SecureFiles/file3.txt

**Questions:**

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

**Student’s answer:**

Added "TestUser" to the group "SecureGroup" by executing: sudo usermod -aG SecureGroup TestUser

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

**Student’s answer:**

"TestUser" couldn't modify "file3.txt" due to lacking write permission for the group (770) on the file.

**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.

**Student’s answer:**

# Enable file access logging for the "SecureFiles" directory

sudo nano /etc/audit/audit.rules

# Add the following line to the audit rules file:

-w /home/username/SecureFiles -p wa

# Restart the audit daemon

sudo systemctl restart auditd

# Create a log file to record all file access events

touch /var/log/file\_access.log

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

cat ~/SecureFiles/file1.txt

cat /var/log/file\_access.log

**Questions:**

How did you enable file access logging for the directory?

**Student’s answer:**

Enabled file access logging by adding a rule in the audit rules file to monitor changes in the "SecureFiles" directory

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

**Student’s answer:**

The log records details like the accessing user, timestamp, and type of access (read/write) when opening "file1.txt" from another account.

4. Conclusions:

**Student’s answer:**

In summary, the experiment demonstrated the importance of implementing file access logging as a proactive security measure in a Linux system. By monitoring and recording file access events, administrators can detect unauthorized activities, track user interactions, and enhance overall system security. This experiment underscores the significance of leveraging file access logging as part of a comprehensive security strategy to safeguard sensitive data and mitigate potential risks.