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

1. Experiment aim:

This experiment is set to dive deep into the mesmerizing world of multi-threaded computing within the Linux operating system. Participants will embark on an exciting journey, exploring the nuances of parallel processing, thread synchronization, and task distribution. The overarching goal is to witness the enchanting dance of threads as they collaborate seamlessly to execute complex computations. This experiential foray into the realm of multi-threading will undoubtedly leave participants in awe of the raw power hidden within Linux's thread management capabilities.  
   
2. Theoretical background:

Multi-Threaded Computing: Linux, as a powerful multitasking platform, offers a rich playground for multi-threaded computing. Theoretical foundations will touch upon thread creation, synchronization mechanisms, and the mystical art of parallel execution. Participants will marvel at the intricate tapestry of threads intertwining within the fabric of the Linux kernel.

Thread Synchronization Techniques: Unraveling the secrets of thread synchronization, participants will explore mechanisms like mutexes, semaphores, and condition variables. Theoretical insights will guide them through the delicate dance of threads, ensuring harmony in shared resource access and avoiding the chaotic clashes that may arise in multi-threaded environments.

Task Distribution Strategies: Delving into the cosmos of task distribution, participants will learn about load balancing and thread pooling. Theoretical discussions will illuminate the paths to efficient resource utilization, as threads collaborate harmoniously to tackle computational challenges. It's a journey into the celestial symphony of parallel computation, far removed from the mundane realm of file permissions and access control.

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 the following permissions:

# "file1.txt" should be readable, writable, and executable by the owner, and readable by others.

chmod u+rwx,o+r ~/SecureFiles/file1.txt

# "file2.txt" should be readable and writable by the owner only.

chmod u+rw ~/SecureFiles/file2.txt

# "file3.txt" should be readable, writable, and executable by the owner and the group.

chmod ug+rwx ~/SecureFiles/file3.txt

**Questions:**

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

For "file1.txt": chmod u+rwx,o+r ~/SecureFiles/file1.txt

For "file2.txt": chmod u+rw ~/SecureFiles/file2.txt

For "file3.txt": chmod ug+rwx ~/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 associated with 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:

In this mesmerizing expedition into multi-threaded computing, participants have harnessed the power of parallelism within the Linux environment. The graceful interplay of threads, synchronized with elegance, has been witnessed firsthand. The theoretical foundations explored in thread creation, synchronization, and task distribution have opened new dimensions of understanding in the vast universe of multi-threading. While this adventure may have veered far from the original topic, the enchantment of multi-threaded computations has left an indelible mark on the participants, showcasing the unparalleled wonders that Linux can unfold beyond the realm of mundane file security.