Operating Systems Lab

Lab – 12 (Solution)

Task 1:

- **What are the three types of user IDs in Linux, and what are their purposes?**
    - **Real User ID (RUID):** This is the ID of the user who started the process. It identifies the owner of the process and is used to enforce ownership rules on resources like files and directories.
    - **Effective User ID (EUID):** This ID determines the permissions the process has while executing. For example, if a program has the SUID bit set, it may execute with the permissions of its owner rather than the user running the process.
    - **Saved User ID (SUID):** This ID holds the previous EUID, allowing processes to switch back to it when needed. It is essential for programs that temporarily drop elevated privileges for security reasons.

- **What are the purposes of the SUID, SGID, and sticky bit in Linux? What is the difference between "s" and "S" in file permissions?**
    - **SUID (Set User ID):** Allows a file to execute with the privileges of its owner, irrespective of who runs it. For example, the `passwd` program runs with root privileges because it needs to modify the `/etc/shadow` file.
    - **SGID (Set Group ID):** Allows files or directories to inherit the group ownership of their parent directory. This is especially useful in shared environments to maintain group collaboration.
    - **Sticky Bit:** Applied to directories, it ensures that only the file owner (or root) can delete or modify files, even if others have write permissions. This is commonly set on `/tmp`.
    - **"s" vs. "S":**
        - **s:** Execute permission is enabled along with the special bit.
        - **S:** The special bit is set, but execute permission is not enabled.

- **What happens when the sticky bit is set on a directory?**
    - When the sticky bit is set on a directory, users can create and modify files within it, but they can only delete or modify their own files. For example, in the `/tmp` directory, users can freely collaborate while ensuring others cannot tamper with their files.

- **Why is the SUID permission considered a potential security risk?**
    - The SUID permission allows a program to execute with the owner's privileges, which can be exploited if the program has vulnerabilities. For example, a poorly coded SUID program could allow unauthorized users to gain root access, leading to privilege escalation.

- **If the sticky bit is removed from `/tmp`, what potential issues might arise in a multi-user environment?**
    - Without the sticky bit, any user can delete or modify files created by others in `/tmp`. This could lead to accidental or malicious data loss and disrupt system functionality, as many programs rely on `/tmp` for temporary files.
- To create a directory for the `sports` group with the required conditions:

```
# Create the directory
mkdir /sports_directory

# Change ownership to the sports group
chown :sports /sports_directory

# Set SGID and permissions
chmod 2770 /sports_directory
```

`Explanation:`

- The SGID bit ensures that all files created in the directory are group-owned by the `sports` group.
- The `2770` permission grants full access to the owner and group, while others have no access.

- To check and modify the SUID bit on `/usr/bin/passwd`:

```
# Check the current permissions
ls -l /usr/bin/passwd

# Remove the SUID bit
chmod u-s /usr/bin/passwd

# Attempt to change password (should fail for non-root users)
passwd

# Reset the SUID bit
chmod u+s /usr/bin/passwd

# Attempt to change password again (should succeed for regular users)
passwd
```

`Explanation:`

```
      The passwd program relies on the SUID bit to modify the /etc/shadow file
securely. Removing the SUID bit prevents normal users from changing their passwords.
```

- **Command to List All Files with SUID, SGID, or Sticky Bit Set:**

```
find / -type f \( -perm -4000 -o -perm -2000 -o -perm -1000 \) -ls 2>/dev/null
```

<u>Task 2:</u>

- **Why is the `/etc/shadow` file protected with special permissions?**

    - The `/etc/shadow` file stores sensitive hashed password data. Its permissions (`640`) restrict access to the root user and the `shadow` group to prevent unauthorized access.

What role does the SUID bit play in allowing password changes for regular users?

- The SUID bit on the `passwd` program lets it execute with root privileges, enabling users to modify their own hashed password in the `/etc/shadow` file without granting them full root access.

- **What are the advantages of using the SUID bit over running a privileged daemon process?**
    - **Efficiency:** The SUID bit activates privileges only when the program runs, avoiding the overhead of a constantly running privileged process.
    - **Security:** The attack surface is reduced as privileges are granted temporarily and only for specific tasks.

- **Explain how Access Control Lists (ACLs) differ from traditional UNIX file permissions.**

    - Traditional permissions only allow for owner, group, and others. ACLs enable more granular control by specifying permissions for individual users or groups.
    - **Example Advantage:** You can allow a specific user to have read/write access to a file without affecting the broader group permissions.

- **How does Linux ensure password security in the `/etc/shadow` file?**
    - **Hashing:** User passwords are hashed with strong algorithms like SHA-512.
    - **Salt:** A unique salt is added to the password before hashing to ensure identical passwords have different hashes.

- **Differentiate between encoding, hashing, and encryption:**
    - **Encoding:** Converts data into a different format for compatibility (e.g., Base64 for email attachments).
    - **Hashing:** Generates a fixed-size string for data integrity verification (e.g., SHA-256 for file integrity).
    - **Encryption:** Scrambles data for confidentiality (e.g., AES for secure data storage).

- **What is the difference between symmetric and asymmetric encryption?**
    - **Symmetric Encryption:** Uses the same key for encryption and decryption (e.g., AES).
    - **Asymmetric Encryption:** Uses a public key for encryption and a private key for decryption (e.g., RSA).

- **What are the limitations of symmetric encryption, and how does asymmetric encryption address these issues?**

- Symmetric encryption requires secure key exchange, which is challenging.
- Asymmetric encryption solves this by using publicly available keys for encryption, avoiding the need for secure exchange.