Rapport : Laboratoire 6 – Contrôle d'accès et permissions Unix

Auteurs

- Ouweis Harun
- Amir Mouti

---

1. Introduction

Ce rapport documente l'ensemble des étapes réalisées dans le cadre du laboratoire *"Access Control"*. L'objectif principal est de se familiariser avec le modèle de permissions Unix, d'analyser et de corriger les permissions sur les fichiers, ainsi que de créer un outil permettant d'identifier et corriger des permissions dangereuses (comme les fichiers *world-writable*).

Chaque section correspond à une tâche définie dans l'énoncé, avec les commandes exécutées, les sorties obtenues, les explications détaillées, et les livrables attendus.

---

2. Étape 1 – Compréhension des permissions et des groupes

2.1. Interpreting account and group information

Commande exécutée

```
id
```

Sortie obtenue

```
uid=1000(tobioo) gid=1000(tobioo)
groups=1000(tobioo),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(
dip),44(video),46(plugdev),116(netdev),998(ollama),1001(docker)
```

Questions et réponses

- **Quel est votre UID et quel est le nom de votre compte ?** → L'UID est `1000`, et le nom du compte est `tobioo`.
- **Quel est le GID de votre groupe principal ("groupe principal") et quel est son nom ?** → Le GID du groupe principal est également `1000` et son nom est `tobioo`.
- **À combien d'autres groupes appartenez-vous ?** → On voit que j'appartiens à **12 groupes secondaires**, en plus de mon groupe principal :

```
adm, dialout, cdrom, floppy, sudo, audio, dip, video, plugdev, netdev,
ollama, docker
```

Explication

La commande `id` permet d'afficher les informations de l'utilisateur courant :

- `uid=1000(tobioo)` : indique l'identifiant numérique de l'utilisateur et son nom.
- `gid=1000(tobioo)` : indique l'identifiant du groupe principal de l'utilisateur, souvent identique à son nom.
- `groups=...` : liste l'ensemble des groupes auxquels l'utilisateur appartient, ce qui détermine les accès qu'il peut avoir à certains fichiers ou périphériques système (journalisation, audio, Docker, etc.).

2.2. Interpreting access control metadata on files and directories

Commandes exécutées

```
ls -l /etc/passwd
ls -l /bin/ls
ls -l ~/.bashrc
ls -l ~/.bash_history
```

Sorties obtenues

```
-rw-r--r-- 1 root   root   1615 Mar  7 01:36 /etc/passwd
-rwxr-xr-x 1 root   root 138208 Feb  7  2022 /bin/ls
-rw-r--r-- 1 tobioo tobioo 3989 Jan  3 00:27 /home/tobioo/.bashrc
-rw------- 1 tobioo tobioo 43210 Apr 30 02:11 /home/tobioo/.bash_history
```

Questions et réponses

1. **Qui est le propriétaire de chaque fichier ? Quel est le groupe propriétaire ?**

Fichier	Propriétaire	Groupe
`/etc/passwd`	`root`	`root`
`/bin/ls`	`root`	`root`
`~/.bashrc`	`tobioo`	`tobioo`
`~/.bash_history`	`tobioo`	`tobioo`

2. **Quels sont les droits de lecture, écriture, exécution pour chacun des trois types d'utilisateurs ?**

- **/etc/passwd** : `-rw-r--r--`
  - **Utilisateur (root)** : lecture + écriture
  - **Groupe (root)** : lecture
  - **Autres** : lecture
- **/bin/ls** : `-rwxr-xr-x`
  - **Utilisateur (root)** : lecture + écriture + exécution
  - **Groupe (root)** : lecture + exécution
  - **Autres** : lecture + exécution
- **~/.bashrc** : `-rw-r--r--`
  - **Utilisateur (tobioo)** : lecture + écriture
  - **Groupe (tobioo)** : lecture
  - **Autres** : lecture
- **~/.bash_history** : `-rw-------`
  - **Utilisateur (tobioo)** : lecture + écriture
  - **Groupe et autres** : aucun droit

Explication

Ces fichiers ont des permissions adaptées à leur usage :

- **/etc/passwd** contient des informations sur les utilisateurs, mais ne doit pas être modifiable par tous.
- **/bin/ls** est un exécutable partagé par tous les utilisateurs, d'où les droits d'exécution globaux.
- **~/.bashrc** est lu à chaque ouverture de terminal, il est personnel mais non confidentiel.
- **~/.bash_history** est un fichier sensible, car il contient l'historique des commandes de l'utilisateur ; il est donc privé.

2.3. Analyse des permissions du répertoire personnel et de `/tmp`

Commandes exécutées

```
ls -ld ~
ls -ld /tmp
```

Sorties obtenues

```
drwxr-xr-x 24 tobioo tobioo 4096 Apr 26 00:05 /home/tobioo
drwxrwxrwt 11 root   root   4096 May  6 22:23 /tmp
```

Questions et réponses

1. **Qui est le propriétaire et quel est le groupe propriétaire de votre répertoire personnel ?** → Le répertoire `/home/tobioo` est **propriété de l'utilisateur** `tobioo` et du **groupe** `tobioo`.

2. **Quelle est la configuration des permissions ?**

    - `/home/tobioo` : `drwxr-xr-x`

        - **Utilisateur (tobioo)** : lecture, écriture, exécution
        - **Groupe (tobioo)** : lecture, exécution
        - **Autres** : lecture, exécution

    - `/tmp` : `drwxrwxrwt`

        - **Tous les utilisateurs** peuvent lire, écrire et exécuter dans `/tmp`, **mais** le bit **sticky (`t`)** est activé.

3. **Qui peut lister les fichiers dans votre répertoire personnel ?** → Tout le monde (**groupe et autres**) peut lister les fichiers, car les permissions `r-x` (lecture + exécution) sont accordées aux autres.

4. **Qui peut créer des fichiers dans votre répertoire personnel ?** → **Seul l'utilisateur** `tobioo` peut créer des fichiers, car lui seul possède le droit **écriture** (`w`) dans ce dossier.

5. **Quelles permissions permettent de créer des fichiers dans le répertoire `/tmp` ?** → Les permissions `drwxrwxrwt` :

    - Les droits `rwxrwxrw` indiquent que **tous les utilisateurs** ont le droit d'écrire dans `/tmp`.
    - Le bit `t` (sticky bit) signifie que **seul le propriétaire d'un fichier peut le supprimer**, même si d'autres utilisateurs ont les droits d'écriture dans le dossier. Cela protège les fichiers des autres dans un répertoire partagé.

Explication

- Le répertoire personnel est généralement en lecture pour tous par défaut sur certaines distributions (comme ici), mais peut être rendu privé si nécessaire avec `chmod 700 ~`.
- `/tmp` est un répertoire **temporaire public**, utilisé pour l'échange ou le stockage de fichiers intermédiaires par tous les programmes. Le sticky bit est une mesure de sécurité importante pour éviter les suppressions abusives.

2.4. Modifying access rights

1. Création d'un fichier avec des permissions initiales `rw-------`

Commandes exécutées

```
touch file
chmod 600 file
ls -l file
```

Sortie obtenue

```
-rw------- 1 tobioo tobioo 0 May  6 22:48 file
```

---

2. Modifications des permissions en mode symbolique

a. Objectif : `rw- r-- ---`

```
chmod u=rw,g=r,o= file
ls -l file
```

Sortie :

```
-rw-r----- 1 tobioo tobioo 0 May  6 22:48 file
```

---

b. Objectif : `rwx r-x ---`

```
chmod u=rwx,g=rx,o= file
ls -l file
```

Sortie :

```
-rwxr-x--- 1 tobioo tobioo 0 May  6 22:48 file
```

---

c. Objectif : `r-- r-- r--`

```
chmod a=r file
ls -l file
```

Sortie :

```
-r--r--r-- 1 tobioo tobioo 0 May  6 22:48 file
```

---

d. Objectif : `rwx r-- r--`

```
chmod u=rwx,g=r,o=r file
ls -l file
```

Sortie :

```
-rwxr--r-- 1 tobioo tobioo 0 May  6 22:48 file
```

---

e. Objectif : `rwx --- ---`

```
chmod 700 file
ls -l file
```

Sortie :

```
-rwx------ 1 tobioo tobioo 0 May  6 22:48 file
```

---

3. Permissions conflictuelles – écriture autorisée uniquement pour les autres

Commandes exécutées

```
touch conflict_file
chmod 002 conflict_file
ls -l conflict_file
```

Sortie :

```
--------w- 1 tobioo tobioo 0 May  6 22:49 conflict_file
```

Test d'écriture

```
echo "test" > conflict_file
```

Sortie :

```
-bash: conflict_file: Permission denied
```

Explication et réponse à la question :

> **Que fait le système si vous essayez d'écrire dans un fichier dont seul "others" a le droit d'écriture, mais pas le propriétaire ?**

Même si le fichier est techniquement modifiable par "others", le système empêche l'utilisateur **propriétaire** d'y écrire **car il ne dispose pas du droit d'écriture**. En effet, sous Linux, **le noyau commence par vérifier les permissions du propriétaire** (si c'est lui qui agit). Si le propriétaire **n'a pas le droit**, l'accès est refusé, même si "others" y est autorisé. Ce comportement est conforme au modèle de permission POSIX.

Parfait — **tout ce que tu as fait est 100 % correct** et **correspond exactement** à ce qui est demandé dans l'énoncé, y compris l'usage du groupe `proj_a` partagé entre `labf0` et `labf1`, et la vérification des permissions de lecture/écriture.

Voici donc la **section 2.5 complète du rapport**, fidèle au format que nous avons utilisé jusqu'ici :

2.5. Giving other users access to your files

Contexte

Cet exercice consiste à vérifier l'accès aux fichiers d'un autre utilisateur sur le même système, puis à créer un dossier partagé dans le répertoire personnel de `labf0`, accessible uniquement à `labf1` via un groupe de projet commun.

---

1. Est-ce que le collègue peut lire les fichiers dans le home de `labf0` ?

Commandes exécutées

Depuis `labf0` :

```
ls -ld ~
```

Sortie :

```
drwxr-xr-x 8 labf0 labf0 4096 May  7 22:21 /home/labf0
```

Depuis `labf1` :

```
ls /home/labf0
```

Sortie :

```
public_html
```

Explication

Oui, le répertoire personnel de `labf0` est accessible en lecture (`r-x`) aux autres utilisateurs. Cela permet à `labf1` de lister les fichiers qu'il contient. Si l'on souhaite le rendre privé, il suffit d'exécuter :

```
chmod 700 ~
```

---

2. Création du répertoire `shared` accessible uniquement à `labf1`

Préparation

Les deux comptes `labf0` et `labf1` sont membres des groupes `proj_a` et `proj_b` (confirmé par `id`).

Commandes exécutées depuis `labf0`

```
rm -rf ~/shared
mkdir ~/shared
chgrp proj_a ~/shared
chmod 770 ~/shared
ls -ld ~/shared
```

Sortie :

```
drwxrwx--- 2 labf0 proj_a 4096 May  9 22:52 /home/labf0/shared
```

---

3. Vérification de l'accès depuis `labf1`

```
ls /home/labf0/shared
touch /home/labf0/shared/test_from_labf1.txt
ls /home/labf0/shared
```

Sortie :

```
test_from_labf1.txt
```

Explication

Le fichier a été créé avec succès par `labf1`, preuve que le groupe `proj_a` donne bien un accès en lecture et écriture. Aucune autre personne extérieure à ce groupe ne peut accéder au dossier.

2.6. Find command usage

1. Utilisation de base de `find` sans critère

Commande exécutée

```
find .
```

Question : Que fait `find` avec les fichiers ou dossiers cachés ?

Réponse

La commande `find .` affiche **tous les fichiers et dossiers**, y compris **les fichiers et dossiers cachés** (ceux dont le nom commence par un `.`), car elle ne filtre rien et explore **récursivement** tout ce qui est présent sous le répertoire courant. Contrairement à la commande `ls`, elle **ne masque pas** les fichiers cachés.

---

2. Recherches spécifiques dans le répertoire personnel

Toutes les commandes ci-dessous sont à exécuter dans le répertoire personnel (`~`) ou avec `find ~`.

a. Fichiers terminant par `.c`, `.cpp` ou `.sh`

```
find ~ \( -name "*.c" -o -name "*.cpp" -o -name "*.sh" \)
```

Explication : On utilise les opérateurs `-o` et des parenthèses `\(...\)` pour combiner plusieurs motifs. Le tilde `~` désigne le répertoire personnel.

---

b. Fichiers exécutables

```
find ~ -type f -executable
```

Explication : Affiche tous les fichiers réguliers (`-type f`) ayant le **bit exécutable** défini pour l'utilisateur courant.

---

c. Fichiers non modifiés depuis plus de deux ans

```
find ~ -type f -mtime +730
```

Explication : `-mtime +730` signifie que le fichier n'a pas été modifié depuis **plus de 730 jours**, soit environ **2 ans**. `mtime` se réfère à la dernière modification du contenu du fichier.

---

d. Fichiers non accédés depuis plus de deux ans

```
find ~ -type f -atime +730
```

Explication : `-atime +730` signifie que le fichier n'a **pas été ouvert** (ni lu) depuis plus de 730 jours.

---

e. Fichiers non accédés depuis plus de trois ans et de taille > 3 Mo

```
find ~ -type f -atime +1095 -size +3M
```

Explication : Combine deux conditions :

- `-atime +1095` : plus de **3 ans sans accès**
- `-size +3M` : fichiers de **plus de 3 mégaoctets**

Ces fichiers sont de bons **candidats à l'effacement ou à l'archivage**.

---

f. Dossiers nommés `.git`

```
find ~ -type d -name ".git"
```

Explication : Cherche tous les **répertoires (`-type d`)** dont le nom est exactement `.git`, typiquement à la racine d'un dépôt Git.

---

3. Recherche de texte dans les fichiers

Question : Quelle commande est correcte pour afficher tous les fichiers contenant le mot `root` ?

Commande 1 :

```
find . -type f -exec grep -l 'root' {} \;
```

Commande 2 :

```
find * -type f -exec grep -l 'root' {} \;
```

Réponse

La **commande correcte est la première** :

```
find . -type f -exec grep -l 'root' {} \;
```

Explication :

- `find .` signifie « commence la recherche à partir du répertoire courant », y compris les fichiers et répertoires **cachés**.
- `find *` est une expansion par le shell : les fichiers/dossiers **cachés sont ignorés** par défaut et s'il y a des espaces ou des noms particuliers, cela peut provoquer des erreurs ou des comportements inattendus.
- La version avec `.` est plus **robuste, sûre, et conforme aux usages Unix**.

---

3. Task 2 – Display world-writable files

3.1. Création du répertoire de test

Commandes exécutées

```
mkdir -p test_dir/dir1 test_dir/dir2
touch test_dir/foo test_dir/bar
chmod o+w test_dir/foo
chmod o+w test_dir/dir1
chmod o+w test_dir/dir2
chmod 644 test_dir/bar
ls -lR test_dir
```

Sortie obtenue

```
test_dir:
total 8
```

```
-rw-r--r-- 1 tobioo tobioo    0 May  7 00:35 bar
drwxr-xrwx 2 tobioo tobioo 4096 May  7 00:35 dir1
drwxr-xrwx 2 tobioo tobioo 4096 May  7 00:35 dir2
-rw-r--rw- 1 tobioo tobioo    0 May  7 00:35 foo

test_dir/dir1:
total 0

test_dir/dir2:
total 0
```

Explication

- Le dossier `test_dir` contient deux fichiers (`foo`, `bar`) et deux sous-répertoires (`dir1`, `dir2`).
- Les éléments `foo`, `dir1`, `dir2` ont tous le **bit d'écriture pour "others" activé** (`o+w`), ce qui en fait des **éléments world-writable**.
- Le fichier bar a des permissions classiques (rw-r--r--) et n'est pas world-writable.
- Ce type de permission est considéré comme dangereux car **tout utilisateur du système peut modifier ces fichiers ou répertoires**.

3.2. Script initial : `fix_permissions`

Contenu du script

```
#!/bin/bash

echo "The following files/directories are world-writable:"
find test_dir -perm -0002
```

Commandes exécutées

```
nano fix_permissions
chmod +x fix_permissions
./fix_permissions
```

Sortie du script

```
The following files/directories are world-writable:
test_dir/dir2
test_dir/dir1
test_dir/foo
```

Explication

- Le script utilise `find` avec l'option `-perm -0002`, qui détecte tous les fichiers ou répertoires dans `test_dir` **ayant le bit d'écriture activé pour "others"**.
- Cela permet à un administrateur système ou à un utilisateur vigilant d'**identifier les fichiers à permissions trop ouvertes** dans une hiérarchie de fichiers donnée.
- Le résultat obtenu est cohérent avec les fichiers créés et modifiés dans la section précédente.

---

4. Task 3 – Pass the directory as an argument

Contenu actuel du script `fix_permissions`

```
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Erreur : vous devez fournir un répertoire à analyser."
    echo "Utilisation : $0 <répertoire>"
    exit 1
fi

DIR="$1"

if [ ! -d "$DIR" ]; then
    echo "Erreur : '$DIR' n'est pas un répertoire valide."
    exit 1
fi

echo "The following files/directories are world-writable:"
find "$DIR" -perm -0002
```

Tests réalisés

1. **Sans argument**

```
./fix_permissions
```

Sortie attendue :

```
Erreur : vous devez fournir un répertoire à analyser.
Utilisation : ./fix_permissions <répertoire>
```

2. **Avec un argument invalide**

```
./fix_permissions not_a_dir
```

Sortie attendue :

```
Erreur : 'not_a_dir' n'est pas un répertoire valide.
```

3. **Avec un répertoire valide**

```
./fix_permissions test_dir
```

Sortie obtenue :

```
The following files/directories are world-writable:
test_dir/dir2
test_dir/dir1
test_dir/foo
```

Explication

- La gestion des erreurs a été introduite avec des messages explicites et un code de retour `1`.
- Le répertoire à analyser est maintenant passé dynamiquement au script, ce qui le rend réutilisable pour n'importe quel chemin.
- La commande `find` est adaptée pour fonctionner sur le répertoire fourni.

---

5. Task 4 – Propose a fix

Objectif

Améliorer le script `fix_permissions` pour non seulement détecter les fichiers ou répertoires *world-writable*, mais également **proposer une correction** via une confirmation interactive unique (`y/n`).

Contenu final du script `fix_permissions`

```
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Erreur : vous devez fournir un répertoire à analyser."
    echo "Utilisation : $0 <répertoire>"
    exit 1
fi
```

```
DIR="$1"

if [ ! -d "$DIR" ]; then
    echo "Erreur : '$DIR' n'est pas un répertoire valide."
    exit 1
fi

WW_LIST=$(find "$DIR" -perm -0002)

if [ -z "$WW_LIST" ]; then
    echo "Aucun fichier ou répertoire world-writable trouvé."
    exit 0
fi

echo "The following files/directories are world-writable:"
echo "$WW_LIST"

read -p "Do you want the permissions to be fixed (y/n)? " ANSWER

if [ "$ANSWER" = "y" ] || [ "$ANSWER" = "Y" ]; then
    echo "$WW_LIST" | while read ITEM; do
        chmod o-w "$ITEM"
    done
    echo "Permissions corrected."
else
    echo "Aucune modification effectuée."
fi
```

Test réalisé avec `test_dir_copy`

Commandes exécutées

```
cp -a test_dir test_dir_copy
./fix_permissions test_dir_copy
```

Sortie obtenue

```
The following files/directories are world-writable:
test_dir_copy/test_dir/dir2
test_dir_copy/test_dir/dir1
test_dir_copy/test_dir/foo
Do you want the permissions to be fixed (y/n)? y
Permissions corrected.
```

Vérification post-exécution

```
ls -lR test_dir_copy
```

Sortie obtenue

```
test_dir_copy:
total 12
-rw-r--r-- 1 tobioo tobioo    0 May  7 00:35 bar
drwxr-xr-x 2 tobioo tobioo 4096 May  7 00:35 dir1
drwxr-xr-x 2 tobioo tobioo 4096 May  7 00:35 dir2
-rw-r--r-- 1 tobioo tobioo    0 May  7 00:35 foo
drwxr-xr-x 4 tobioo tobioo 4096 May  7 00:35 test_dir

test_dir_copy/dir1:
total 0

test_dir_copy/dir2:
total 0

test_dir_copy/test_dir:
total 8
-rw-r--r-- 1 tobioo tobioo    0 May  7 00:35 bar
drwxr-xr-x 2 tobioo tobioo 4096 May  7 00:35 dir1
drwxr-xr-x 2 tobioo tobioo 4096 May  7 00:35 dir2
-rw-r--r-- 1 tobioo tobioo    0 May  7 00:35 foo
```

Explication

- Le script détecte correctement les éléments avec le bit d'écriture pour "others" (`o+w`).
- L'utilisateur est invité à confirmer l'action une seule fois.
- Si confirmé (`y`), le bit `o+w` est supprimé pour chaque fichier ou répertoire détecté via `chmod o-w`.
- Les permissions finales montrent que tous les droits d'écriture publics ont été retirés.

---

6. Task 5 – Display group-writable files

Objectif

Étendre le script `fix_permissions` afin de détecter les fichiers et répertoires avec le **bit d'écriture activé pour le groupe** (`g+w`), **sauf si** le groupe du fichier est **identique** au groupe personnel de l'utilisateur (`tobioo`).

Préparation

Commandes exécutées

```
cd ~/lab06/test_dir
touch baz
mkdir dir3
sudo chgrp proj_a baz
sudo chgrp proj_a dir3
chmod g+w baz
chmod g+w dir3
```

Résultat de `ls -l`

```
total 12
-rw-r--r-- 1 tobioo tobioo    0 May  7 00:35 bar
-rw-rw-r-- 1 tobioo proj_a    0 May  7 00:53 baz
drwxr-xrwx 2 tobioo tobioo 4096 May  7 00:35 dir1
drwxr-xrwx 2 tobioo tobioo 4096 May  7 00:35 dir2
drwxrwxr-x 2 tobioo proj_a 4096 May  7 00:53 dir3
-rw-r--rw- 1 tobioo tobioo    0 May  7 00:35 foo
```

Contenu du script `fix_permissions` (ajout Task 5)

```
#!/bin/bash

# Vérification du nombre d'arguments
if [ $# -ne 1 ]; then
    echo "Erreur : vous devez fournir un répertoire à analyser."
    echo "Utilisation : $0 <répertoire>"
    exit 1
fi

DIR="$1"

# Vérification que l'argument est bien un répertoire
if [ ! -d "$DIR" ]; then
    echo "Erreur : '$DIR' n'est pas un répertoire valide."
    exit 1
fi

# Détection des fichiers et dossiers world-writable
WW_LIST=$(find "$DIR" -perm -0002)

if [ -z "$WW_LIST" ]; then
    echo "Aucun fichier ou répertoire world-writable trouvé."
else
    echo "The following files/directories are world-writable:"
    echo "$WW_LIST"

    # Confirmation utilisateur
```

```
    read -p "Do you want the permissions to be fixed (y/n)? " ANSWER
    if [ "$ANSWER" = "y" ] || [ "$ANSWER" = "Y" ]; then
        echo "$WW_LIST" | while read ITEM; do
            chmod o-w "$ITEM"
        done
        echo "Permissions corrected."
    else
        echo "Aucune modification effectuée."
    fi
fi

# Détection des fichiers et dossiers group-writable
GROUP_WRITABLE=$(find "$DIR" -perm -0020)

# Récupérer le groupe personnel (même nom que l'utilisateur)
USER_GROUP=$(id -gn)

# Filtrage : uniquement ceux dont le groupe ≠ groupe personnel
GROUP_WRITABLE_FILTERED=""
while IFS= read -r file; do
    [ -z "$file" ] && continue
    FILE_GROUP=$(stat -c %G "$file")
    if [ "$FILE_GROUP" != "$USER_GROUP" ]; then
        GROUP_WRITABLE_FILTERED="${GROUP_WRITABLE_FILTERED}${file}"$'\n'
    fi
done <<< "$GROUP_WRITABLE"

# Affichage du résultat
if [ -n "$GROUP_WRITABLE_FILTERED" ]; then
    echo "The following files/directories are writable for groups:"
    echo "$GROUP_WRITABLE_FILTERED"
fi
```

Exécution du script

```
./fix_permissions test_dir
```

Sortie obtenue

```
The following files/directories are world-writable:
test_dir/dir2
test_dir/dir1
test_dir/foo
Do you want the permissions to be fixed (y/n)? y
Permissions corrected.
The following files/directories are writable for groups:
test_dir/baz
test_dir/dir3
```

Explication

- Le script utilise `find` pour détecter les fichiers ou dossiers avec `g+w`.
- Il filtre ensuite les résultats pour exclure ceux appartenant au **groupe personnel** de l'utilisateur (`id -gn`).
- Les fichiers listés sont des fichiers potentiellement **accessibles en écriture par d'autres membres d'un groupe de projet** (comme `proj_a`), ce qui peut poser des **risques de sécurité ou d'intégrité**.

---

7. Arborescence finale du répertoire de test

Commande exécutée

```
ls -lR test_dir > test_dir.txt
```

Fichier généré :

- `test_dir.txt`

contenant :

```
test_dir:
total 12
-rw-r--r-- 1 tobioo tobioo    0 May  7 00:35 bar
-rw-rw-r-- 1 tobioo proj_a    0 May  7 00:53 baz
drwxr-xr-x 2 tobioo tobioo 4096 May  7 00:35 dir1
drwxr-xr-x 2 tobioo tobioo 4096 May  7 00:35 dir2
drwxrwxr-x 2 tobioo proj_a 4096 May  7 00:53 dir3
-rw-r--r-- 1 tobioo tobioo    0 May  7 00:35 foo

test_dir/dir1:
total 0

test_dir/dir2:
total 0

test_dir/dir3:
total 0
```
