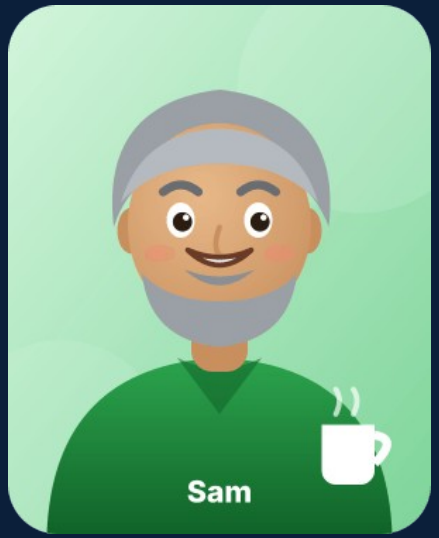
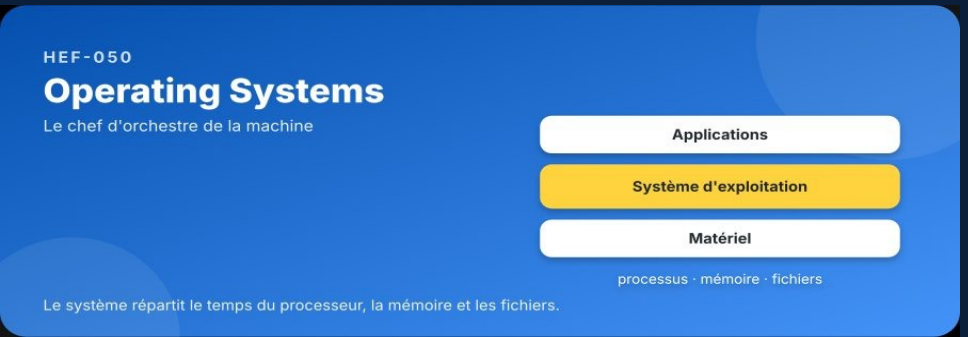
HEF-050 · MISSION 07 / 10

# Permissions et identité

*Qui es-tu ? Qu'as-tu le droit de faire ? Deux questions à la base de toute sécurité.*




**Badge : Gardien des accès**

70 min · intermédiaire

# L'histoire

Un script de HashBox plante : « Permission denied ». Kodi lance `sudo chmod 777` sur le dossier entier « pour que ça marche ». Le lendemain, un audit remarque que n'importe quel utilisateur de la machine peut modifier les fichiers de configuration.

> *« Tu as résolu ton problème en supprimant la protection, dit Sam. Apprenons à donner **exactement** les droits nécessaires. »*




**Tu maîtrises le modèle d'identité et de permissions des systèmes Unix.**

Ta mission

# À la fin, je serai capable de…

1. lire et écrire des permissions rwx en notation symbolique et octale
2. distinguer propriétaire, groupe et autres
3. expliquer le rôle de `umask`
4. expliquer le principe du moindre privilège
5. expliquer pourquoi `chmod 777` est presque toujours une mauvaise réponse

Défi d'échauffement

# Défi d'échauffement

Dans un bureau : le directeur peut lire, modifier et supprimer le dossier « Salaires ». Les RH peuvent le lire. Les autres employés n'ont aucun accès. Écris une règle pour trois catégories de personnes et trois types d'actions (lire, écrire, entrer dans le bureau). Comment représenterais-tu cette règle dans un tableau de 3 × 3 ?

Le concept

# Identité : utilisateur et groupes

Chaque processus s'exécute pour un **utilisateur** (UID) et des **groupes** (GID). Le super-utilisateur **root** (UID 0) contourne la plupart des contrôles.

## Les trois droits

| Droit | Sur un fichier | Sur un dossier |
|---|---|---|
| **r** (read) | lire le contenu | lister les noms |
| **w** (write) | modifier le contenu | créer, renommer, supprimer des entrées |
| **x** (execute) | exécuter | **traverser** (entrer dans) le dossier |

Le concept

# Les trois catégories

Chaque fichier a un **propriétaire**, un **groupe** et des droits pour trois catégories :

```
-rwxr-x---   1 sam equipe   1204 mars 12 deploy.sh
 |└─┘└─┘└─┘
 | |  |  |  └── autres : aucun droit
 | |  |  └───── groupe : r-x (lire, exécuter)
 | |  └──────── propriétaire : rwx
 | └─────────── type (- fichier, d dossier, l lien)
```

## Notation octale

Chaque droit vaut un chiffre : **r = 4, w = 2, x = 1**. On additionne pour chaque catégorie.

Le concept

# Notation octale (suite)

| Symbolique | Octal |
|---|---|
| `rwx` | 7 |
| `rw-` | 6 |
| `r-x` | 5 |
| `r--` | 4 |
| `---` | 0 |

`rwxr-x---` = **750**. `rw-r--r--` = **644**. `rwx------` = **700**.

## Commandes essentielles

Le concept

# Commandes essentielles (suite)

```
chmod 640 config.ini        # rw-r-----
chmod u+x script.sh          # ajoute x pour le propriétaire
chown sam:equipe fichier     # change propriétaire et groupe
id                           # affiche mon UID et mes groupes
```

## umask

Le **umask** indique quels droits **retirer** à la création d'un fichier. Avec `umask 022`, un nouveau fichier (base 666) devient 644 et un nouveau dossier (base 777) devient 755.

## Le moindre privilège

> *Donner **uniquement** les droits nécessaires, pour le temps nécessaire.*

- un service web n'a pas besoin d'être `root` ;

Le concept

# Le moindre privilège (suite)

- un fichier de mots de passe est en `600` ;
- un script exécutable n'a pas besoin d'être modifiable par tous.

## Droits spéciaux

- **setuid** : le programme s'exécute avec les droits de son **propriétaire** (par exemple `passwd`). Puissant et risqué.
- **sticky bit** sur un dossier (`/tmp`) : seuls le propriétaire du fichier et root peuvent le supprimer.

## Avec Python

Le concept

# Avec Python (suite)

```
import os, stat, tempfile

with tempfile.TemporaryDirectory() as d:
    p = os.path.join(d, "secret.txt")
    open(p, "w").write("x")
    os.chmod(p, 0o640)
    print(stat.filemode(os.stat(p).st_mode))       # -rw-r-----
    print(oct(stat.S_IMODE(os.stat(p).st_mode)))   # 0o640
```

Exemple pas à pas

# Exemple pas à pas

**Problème** : traduire des droits entre notation octale et symbolique, et vérifier un accès.

```
def octal_vers_symbolique(mode):
    texte = ""
    for chiffre in f"{mode:03o}":
        n = int(chiffre)
        texte += ("r" if n & 4 else "-") + ("w" if n & 2 else "-") + ("x" if n & 1
    else "-")
    return texte

def a_le_droit(mode, categorie, droit):
    # categorie : 0 propriétaire, 1 groupe, 2 autres ; droit : 4 (r), 2 (w), 1 (x)
    chiffre = int(f"{mode:03o}"[categorie])
    return chiffre & droit != 0

print(octal_vers_symbolique(0o750))     # rwxr-x---
print(octal_vers_symbolique(0o644))     # rw-r--r--
print(a_le_droit(0o640, 1, 4))          # True : le groupe peut lire
```

Exemple pas à pas

# Exemple pas à pas (suite)

```
print(a_le_droit(0o640, 2, 4))          # False : les autres ne peuvent pas lire
```

**Cas concret** : un fichier de configuration `config.ini` contient un mot de passe de base de données.

| Droits | Verdict |
|---|---|
| `777` | n'importe qui peut lire et modifier : **inacceptable** |
| `644` | tout le monde peut lire : **fuite du mot de passe** |
| `640` (propriétaire = service, groupe = admins) | le service lit/écrit, les admins lisent, les autres rien : **correct** |
| `600` | le plus strict : seul le propriétaire : **le plus sûr** si les admins n'ont pas besoin de lire |

**Dossier** : pour qu'un utilisateur lise `/projet/donnees/fichier`, il lui faut `x` sur `/projet` et `/projet/donnees`, puis `r` sur `fichier`.

À toi, avec aide

# À toi, avec aide

Complète le tableau :

| Notation octale | Notation symbolique | Qui peut faire quoi ? |
|---|---|---|
| 755 | ? | ? |
| 600 | ? | ? |
| ? | `rw-rw-r--` | ? |
| 400 | ? | ? |

Puis écris un programme qui vérifie tes réponses.

À toi, avec aide · réponse

# Indices progressifs

## Indice 1

7 = 4+2+1, 5 = 4+1, 6 = 4+2.

## Indice 2

Pour `rw-rw-r--` : rw- = 6, rw- = 6, r-- = 4.

## Indice 3

400 : lecture seule pour le propriétaire, rien pour les autres.

À toi, avec aide · réponse

# Réponse

| Octal | Symbolique | Qui peut faire quoi ? |
|---|---|---|
| 755 | `rwxr-xr-x` | propriétaire : tout ; groupe et autres : lire et exécuter |
| 600 | `rw-------` | propriétaire : lire/écrire ; personne d'autre |
| 664 | `rw-rw-r--` | propriétaire et groupe : lire/écrire ; autres : lire |
| 400 | `r--------` | propriétaire : lecture seule |

```
import stat

for mode in (0o755, 0o600, 0o664, 0o400):
    print(oct(mode), stat.filemode(stat.S_IFREG | mode))
```

# À toi, en solo

**Audit de permissions.** Écris `auditer(dossier)` qui parcourt un dossier et signale :

1. les fichiers lisibles par « tous » dont le nom contient `secret`, `password`, `.env` ou `key` ;
2. les fichiers modifiables par « tous » (droit w pour les autres) ;
3. les dossiers sans `x` pour leur propriétaire.

Renvoie une liste de `(chemin, problème)`. Teste en créant des fichiers de test avec `os.chmod`.

À toi, en solo · réponse

# Pistes de correction

```
import os, stat, tempfile

MOTS = ("secret", "password", ".env", "key")

def auditer(dossier):
    problemes = []
    for racine, dossiers, fichiers in os.walk(dossier):
        for nom in dossiers:
            p = os.path.join(racine, nom)
            if not os.stat(p).st_mode & stat.S_IXUSR:
                problemes.append((p, "dossier sans x pour le propriétaire"))
        for nom in fichiers:
            p = os.path.join(racine, nom)
            mode = os.stat(p).st_mode
            if mode & stat.S_IROTH and any(m in nom.lower() for m in MOTS):
                problemes.append((p, "fichier sensible lisible par tous"))
            if mode & stat.S_IWOTH:
                problemes.append((p, "modifiable par tous"))
    return problemes
```

À toi, en solo · réponse

# Pistes de correction (suite)

```
with tempfile.TemporaryDirectory() as d:
    for nom, mode in (("config.ini", 0o640), ("secret.key", 0o644), ("script.sh",
   0o777)):
        p = os.path.join(d, nom)
        open(p, "w").write("x")
        os.chmod(p, mode)
    for chemin, probleme in auditer(d):
        print(os.path.basename(chemin), "→", probleme)
```

Attendu : `secret.key` signalé (sensible lisible), `script.sh` signalé (modifiable par tous). C'est le principe des outils d'audit automatiques.

# Pièges classiques

| Piège | Pourquoi c'est un problème | Comment l'éviter |
|---|---|---|
| **`chmod 777` pour « que ça marche »** | N'importe qui peut modifier | Chercher le droit manquant exact |
| **Fichier secret en 644** | Tout le monde peut lire le mot de passe | 600 ou 640 selon le besoin |
| **Oublier `x` sur un dossier** | Impossible d'y entrer malgré r | x = traverser |
| **Exécuter un service en root** | Une faille donne tout | Utilisateur dédié avec moindre privilège |

Mini-boss

# Mini-boss

**Plan de droits pour HashBox.** L'arborescence :

```
/srv/hashbox/
├── app/            (code, lu par le service web)
├── config/         (fichiers avec mots de passe)
├── donnees/        (uploads des utilisateurs, écrits par le service)
└── sauvegardes/    (copies, écrites par un script de sauvegarde)
```

Utilisateurs : `web` (service), `backup` (sauvegarde), `admin` (humain), groupe `equipe`.

1. Propose propriétaire, groupe et droits (octal) pour chaque dossier et ses fichiers, avec justification (moindre privilège).
2. Écris un script Python qui **vérifie** une arborescence réelle (fictive dans un dossier temporaire) par rapport à ton plan et signale les écarts.
3. Explique ce qui se passe si `web` est compromis : que peut-il lire/écrire ?

Mini-boss · réponse

# Pistes de correction

Exemple de plan : `app/` propriétaire `admin`, groupe `equipe`, 755 (le service lit et exécute, ne modifie pas) ; `config/` propriétaire `web`, 700, fichiers 600 ; `donnees/` propriétaire `web`, 750 (le groupe `backup` ou `equipe` lit) ; `sauvegardes/` propriétaire `backup`, 700. Si `web` est compromis : il peut modifier `donnees/` et lire `config/` (donc les mots de passe) mais ne peut pas modifier `app/` (pas de persistance dans le code) ni lire `sauvegardes/`. Le script de vérification compare `stat.S_IMODE` à un dictionnaire de modes attendus par chemin et liste les écarts.

Quiz éclair

# Quiz éclair

**1.** Les droits `rwxr-x---` correspondent en octal à :

A. 750
B. 755
C. 640

**2.** Sur un dossier, le droit `x` permet de :

A. l'exécuter comme un programme
B. le traverser (y entrer)
C. le supprimer

**3.** Le principe du moindre privilège dit de :

A. donner tous les droits aux administrateurs

Quiz éclair

# Quiz éclair (suite)

B. ne jamais utiliser de mot de passe
C. donner seulement les droits nécessaires

**4.** 🔁 *Révision mission 06* : Un lien physique :

A. est un deuxième nom pour le même inode
B. contient un chemin
C. ne peut exister qu'une fois

**5.** 🔁 *Révision mission 06* : Un inode contient :

A. le nom du fichier
B. les métadonnées du fichier
C. le contenu en clair

Quiz éclair · réponse

# Réponses

1. **A**.
2. **B**.
3. **C**.
4. **A**.
5. **B**.

# Je réfléchis

1. Quels droits d'accès donnes-tu (ou ne donnes-tu pas) à d'autres personnes dans ta vie numérique (partage de documents, comptes) ?
2. Pourquoi `chmod 777` est-il comparable à laisser sa porte d'entrée grande ouverte « pour ne pas perdre la clé » ?

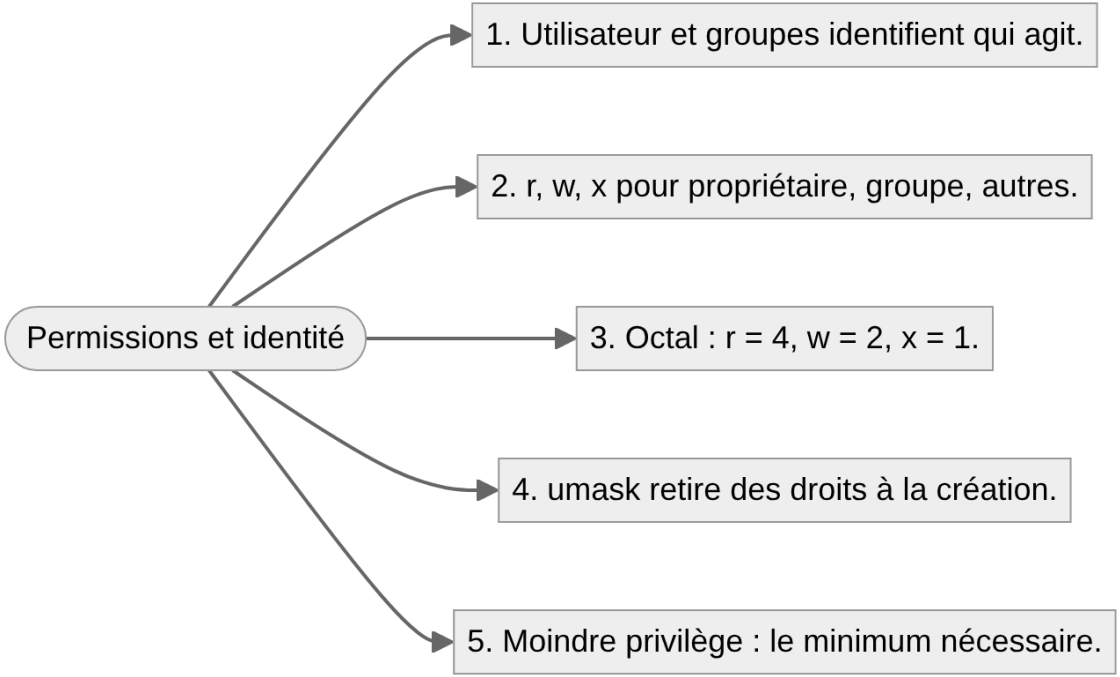
À retenir

# À retenir

1. Utilisateur et groupes identifient qui agit.
2. r, w, x pour propriétaire, groupe, autres.
3. Octal : r = 4, w = 2, x = 1.
4. umask retire des droits à la création.
5. Moindre privilège : le minimum nécessaire.

À retenir

# Le schéma de la mission



MISSION ACCOMPLIE

# +100 XP

## Badge débloqué : Gardien des accès

Prochaine mission : Entrées/sorties : tout est fichier

