BTS SIO · OPTION SISR · SESSION 2026

BAGARRY Matthias

DOSSIER TECHNIQUE

Mise en œuvre, sécurisation et exploitation d'un serveur Linux de gestion de badges sous Ubuntu 24.04

---

0 CONTEXTE ET OBJECTIFS

Ce projet a été réalisé dans le cadre d'une mise en situation professionnelle pour le compte fictif de la société FormaTech, organisme de formation technique de 34 salariés répartis sur deux sites (Paris et Marseille). L'intervention couvre la mise en place du serveur Linux chargé de centraliser les logs du système d'authentification par badge. L'objectif n'était pas une mise en production réelle mais la construction d'un environnement complet couvrant quatre compétences fondamentales de l'administration système Linux.

▸ OBJECTIFS PÉDAGOGIQUES

Maîtriser le cloisonnement des droits d'accès via la double mécanique POSIX (chmod, chown, chgrp) et ACL étendues (setfacl, getfacl), déployer un service SSH durci pour l'administration à distance, mettre en place un dispositif anti-force-brute (Fail2Ban), et valider l'ensemble par une phase de recette rigoureuse avant livraison.

▸ POURQUOI UN SERVEUR DÉDIÉ AUX LOGS DE BADGES

Dans une infrastructure d'entreprise, les systèmes de contrôle d'accès physiques génèrent des événements critiques : horodatage des entrées-sorties, tentatives refusées, identification des agents. Ces logs doivent être centralisés sur un serveur distinct du contrôleur de badges pour garantir l'intégrité en cas de compromission, permettre l'audit par des équipes distinctes (technique et informatique), et survivre à une panne du contrôleur. Le serveur FormaTech joue ce rôle de dépôt de logs, avec des droits d'accès différenciés selon le profil métier.

1 ANALYSE ET ARCHITECTURE

Le projet consiste à déployer un serveur Ubuntu 24.04.3 LTS organisant l'arborescence de fichiers selon les droits attendus par la maîtrise d'ouvrage, sécurisant l'accès distant et prévenant les attaques par force brute sur le service SSH.

1.1 Contraintes fonctionnelles

Élément	Exigence
Système d'exploitation	Ubuntu Server 24.04.3 LTS
Groupes métier	info (informatique) et tech (technique)
Utilisateurs	3 comptes par groupe + compte maître badg0
Accès distant	SSH fonctionnel, root interdit
Protection	Fail2Ban sur SSH, bannissement après 5 échecs
/FORMATECH/LOG	Lecture pour info et tech, écriture pour badg0
/FORMATECH/SCRIPT	Lecture/écriture pour info et badg0, rien pour tech

1.2 Cartographie des droits attendus

▸ CONTRAINTE ARCHITECTURALE

Un dossier ne peut appartenir qu'à un seul groupe propriétaire en UNIX standard. Or le dossier LOG doit être accessible en lecture aux deux groupes info et tech simultanément. La seule solution propre est d'utiliser les ACL POSIX étendues (paquet acl, commandes setfacl / getfacl) qui permettent d'assigner des droits à plusieurs groupes sur un même inode.

Ressource	badg0 (maître)	groupe info	groupe tech
/FORMATECH/LOG	rwx (propriétaire)	r-x (ACL)	r-x (ACL)
/FORMATECH/SCRIPT	rwx (propriétaire)	rwx (groupe prop.)	aucun accès

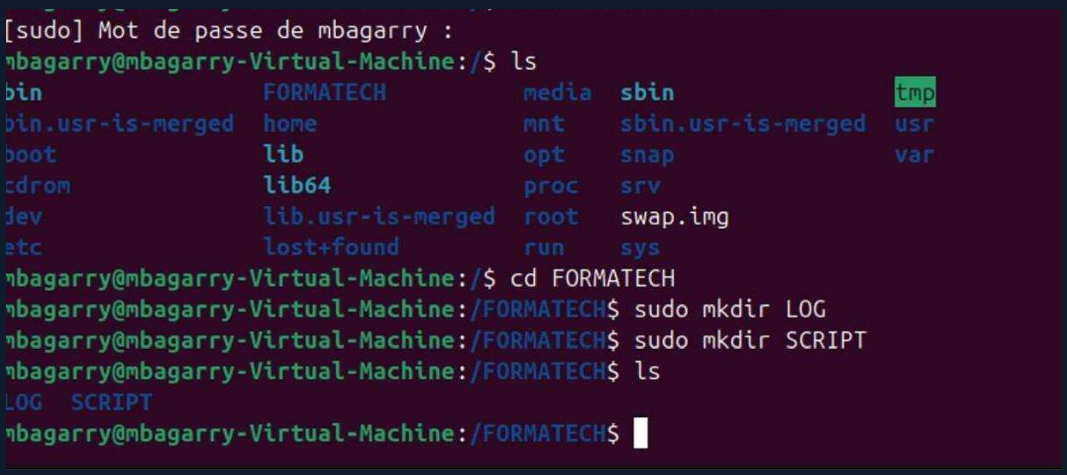
2 ARBORESCENCE ET PERMISSIONS

2.1 Création de l'arborescence

Le dossier racine /FORMATECH est créé à la racine du système pour isoler les données métier du reste du système de fichiers. Deux sous-dossiers concentrent les données : LOG pour les journaux de badgeage, SCRIPT pour les scripts d'exploitation du système de badges.

```
# Création de la racine projet
sudo mkdir /FORMATECH

# Sous-dossiers métier
sudo mkdir /FORMATECH/LOG
sudo mkdir /FORMATECH/SCRIPT
```



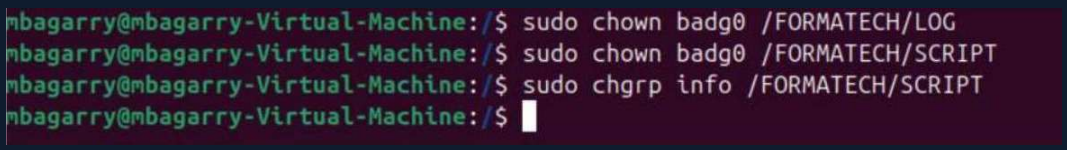
Création de l'arborescence /FORMATECH avec LOG et SCRIPT

2.2 Attribution du propriétaire et du groupe propriétaire

Le compte badg0 (compte maître destiné au système de badges) devient propriétaire des deux dossiers. Le groupe info est positionné comme groupe propriétaire de SCRIPT, ce qui donne automatiquement à ses membres les droits de groupe sur ce dossier.

```
# badg0 devient propriétaire des deux dossiers
sudo chown badg0 /FORMATECH/LOG
sudo chown badg0 /FORMATECH/SCRIPT

# Le groupe info devient groupe propriétaire de SCRIPT
sudo chgrp info /FORMATECH/SCRIPT
```



Attribution du propriétaire (chown) et du groupe (chgrp)

2.3 Droits de base via chmod

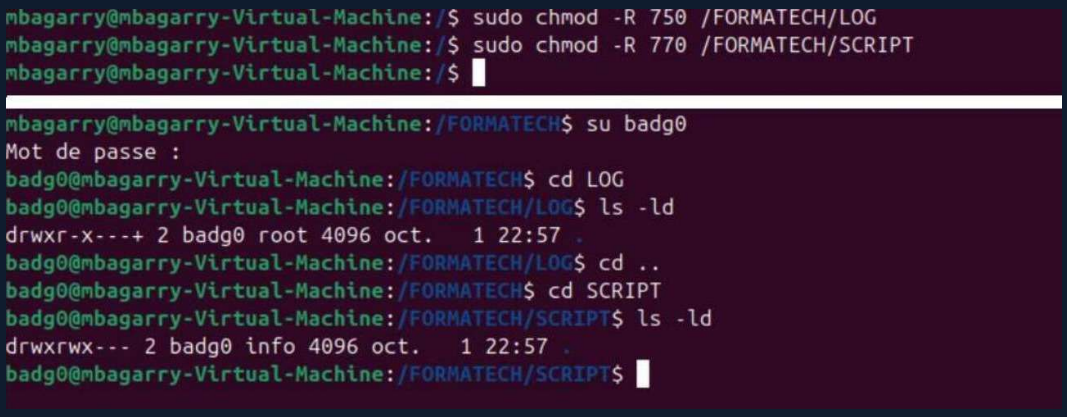
Les permissions octales sont appliquées récursivement (-R) pour couvrir le contenu futur des dossiers. LOG reçoit 750 (rwx propriétaire, r-x groupe, rien pour les autres) et SCRIPT reçoit 770 (rwx propriétaire, rwx groupe, rien pour les autres).

```
sudo chmod -R 750 /FORMATECH/LOG
sudo chmod -R 770 /FORMATECH/SCRIPT

# Vérification avec ls -ld
ls -ld /FORMATECH/LOG /FORMATECH/SCRIPT
```

▸ **LECTURE DU CHMOD 750 ET 770**

7 = 4+2+1 = rwx (lecture + écriture + exécution). 5 = 4+0+1 = r-x (lecture + exécution, pas d'écriture). 0 = aucun droit. Sur un dossier, le bit x signifie traverser : sans lui, impossible de faire cd dedans même avec la lecture. C'est pour cela que les groupes r-x ont bien le droit de lister et d'entrer dans LOG.



Application des chmod et vérification via ls -ld sous le compte badg0

2.4 Partage de LOG via ACL POSIX

Le mode UNIX classique ne permet qu'un seul groupe propriétaire. Pour donner simultanément les droits r-x aux groupes info et tech sur LOG, on superpose les ACL étendues au-dessus des permissions traditionnelles. La commande setfacl pose les ACL, getfacl les relit.

```
# Ajout des ACL pour info et tech en lecture/exécution sur LOG
sudo setfacl -m g:info:rx,g:tech:rx /FORMATECH/LOG

# Vérification complète (droits UNIX + ACL)
getfacl /FORMATECH/LOG
getfacl /FORMATECH/SCRIPT
```

▸ POURQUOI LE « + » DANS LS -L

Après application des ACL, ls -ld affiche un « + » en fin de bloc de droits (ex : drwxr-x---+). Ce symbole indique la présence d'ACL étendues au-delà des droits UNIX visibles. Sans getfacl, impossible de voir quels groupes ou utilisateurs sont réellement autorisés : c'est un point d'attention pour l'audit.

```
mbagarry@mbagarry-Virtual-Machine:/FORMATECH$ sudo setfacl -m g:info:rx,g:tech:rx LO

mbagarry@mbagarry-Virtual-Machine:/FORMATECH$ getfacl LOG
# file: LOG
# owner: badg0
# group: root
user::rwx
group::r-x
group:tech:r-x
group:info:r-x
mask::r-x
other::r-x

mbagarry@mbagarry-Virtual-Machine:/FORMATECH$ getfacl SCRIPT
# file: SCRIPT
# owner: badg0
# group: info
user::rwx
group::r-x
other::r-x

mbagarry@mbagarry-Virtual-Machine:/FORMATECH$
```

Application des ACL sur LOG puis vérification avec getfacl

3 COMPTES UTILISATEURS ET GROUPES MÉTIER

Sept comptes au total sont créés : six comptes nominatifs badg1 à badg6 répartis dans les groupes info et tech selon la position géographique des salariés, et un compte maître badg0 dédié au système de badges lui-même. Ce compte reçoit la propriété des dossiers métier.

3.1 Création des comptes utilisateurs

La commande adduser, contrairement à useradd, est interactive : elle crée automatiquement le répertoire personnel dans /home, copie le squelette /etc/skel et demande le mot de passe. Elle est préférable pour des comptes utilisateurs standards.

```
# Création des 6 utilisateurs nominatifs
sudo adduser badg1
sudo adduser badg2
sudo adduser badg3
sudo adduser badg4
sudo adduser badg5
sudo adduser badg6

# Création du compte maître
sudo adduser badg0
```

▸ **ADDUSER VERSUS USERADD**

useradd est l'outil système bas-niveau : il crée juste l'entrée dans /etc/passwd sans home ni mot de passe. adduser est un wrapper Debian/Ubuntu plus complet qui gère l'ensemble de la procédure (home, skel, shell, mot de passe, GECOS). Dans un contexte d'administration manuelle comme ici, adduser est la bonne commande ; useradd reste utile dans les scripts d'automatisation.

```
mbagarry@mbagarry-Virtual-Machine:/FORMATECH$ sudo adduser badg1
[sudo] Mot de passe de mbagarry :
info: Ajout de l'utilisateur « badg1 » ...
info: Choix d'un UID/GID dans la plage 1000 à 59999 ...
info: Ajout du nouveau groupe « badg1 » (1001) ...
info: Ajout du nouvel utilisateur « badg1 » (1001) avec le groupe « badg1 » (100
1) ...
info: Création du répertoire personnel « /home/badg1 » ...
info: Copie des fichiers depuis « /etc/skel » ...
Nouveau mot de passe :
MOT DE PASSE INCORRECT : Le mot de passe comporte moins de 8 caractères
Retapez le nouveau mot de passe :
passwd : mot de passe mis à jour avec succès
Modifier les informations associées à un utilisateur pour badg1
Entrer la nouvelle valeur, ou appuyer sur ENTER pour la valeur par défaut
        NOM []:
        Numéro de chambre []:
        Téléphone professionnel []:
        Téléphone personnel []:
        Autre []:
Ces informations sont-elles correctes ? [O/n] o
info: Ajout du nouvel utilisateur « badg1 » aux groupes supplémentaires « users
```

Création interactive de l'utilisateur badg1 avec adduser

```
mbagarry@mbagarry-Virtual-Machine:/FORMATECH$ sudo adduser badg0
info: Ajout de l'utilisateur « badg0 » ...
info: Choix d'un UID/GID dans la plage 1000 à 59999 ...
info: Ajout du nouveau groupe « badg0 » (1007) ...
info: Ajout du nouvel utilisateur « badg0 » (1007) avec le groupe « badg0 » (100
7) ...
info: Création du répertoire personnel « /home/badg0 » ...
info: Copie des fichiers depuis « /etc/skel » ...
Nouveau mot de passe :
MOT DE PASSE INCORRECT : Le mot de passe comporte moins de 8 caractères
Retapez le nouveau mot de passe :
passwd : mot de passe mis à jour avec succès
Modifier les informations associées à un utilisateur pour badg0
Entrer la nouvelle valeur, ou appuyer sur ENTER pour la valeur par défaut
        NOM []:
        Numéro de chambre []:
        Téléphone professionnel []:
        Téléphone personnel []:
        Autre []:
Ces informations sont-elles correctes ? [O/n] o
info: Ajout du nouvel utilisateur « badg0 » aux groupes supplémentaires « users
```

Création du compte maître badg0 pour le système de badges

3.2 Vérification dans /etc/passwd

Le fichier /etc/passwd liste tous les comptes du système. Chaque ligne décrit un utilisateur au format nom:x:UID:GID:GECOS:home:shell. Les UID 1000 et plus correspondent aux comptes humains (ici 1001 à 1007), les UID inférieurs aux comptes système.

```
saned:x:113:116::/var/lib/saned:/usr/sbin/nologin
geoclue:x:114:117::/var/lib/geoclue:/usr/sbin/nologin
cups-browsed:x:115:114::/nonexistent:/usr/sbin/nologin
hplip:x:116:7:HPLIP system user,,,:/run/hplip:/bin/false
gnome-remote-desktop:x:988:988:GNOME Remote Desktop:/var/lib/gnome-remote-deskto
p:/usr/sbin/nologin
polkitd:x:987:987:User for polkitd:/:/usr/sbin/nologin
rtkit:x:117:119:RealtimeKit,,,:/proc:/usr/sbin/nologin
colord:x:118:120:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/no
login
gnome-initial-setup:x:119:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:120:121:Gnome Display Manager:/var/lib/gdm3:/bin/false
nm-openvpn:x:121:122:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin
/nologin
mbagarry:x:1000:1000:mbagarry:/home/mbagarry:/bin/bash
sshd:x:122:65534::/run/sshd:/usr/sbin/nologin
badg1:x:1001:1001:,,,:/home/badg1:/bin/bash
badg2:x:1002:1002:,,,:/home/badg2:/bin/bash
badg3:x:1003:1003:,,,:/home/badg3:/bin/bash
badg4:x:1004:1004:,,,:/home/badg4:/bin/bash
badg5:x:1005:1005:,,,:/home/badg5:/bin/bash
badg6:x:1006:1006:,,,:/home/badg6:/bin/bash
badg0:x:1007:1007:,,,:/home/badg0:/bin/bash
mbagarry@mbagarry-Virtual-Machine:/FORMATECH$
```

Listing complet des comptes via cat /etc/passwd : badg0 à badg6 visibles

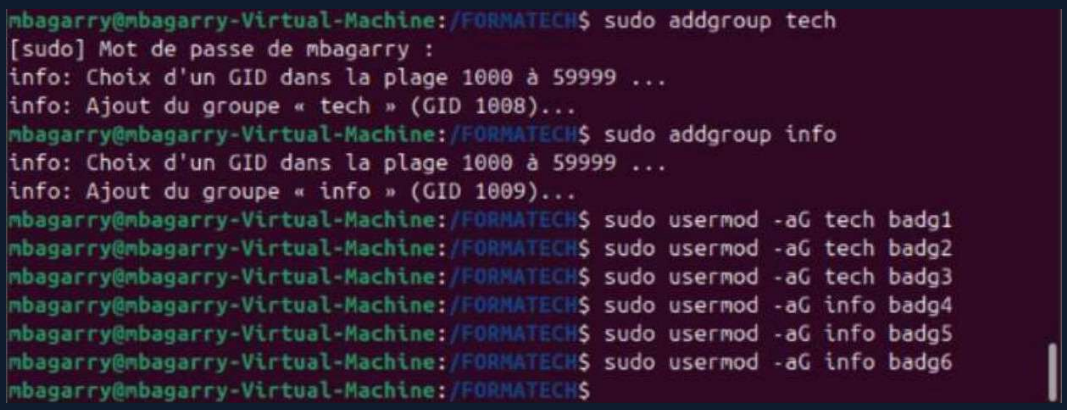
3.3 Création des groupes et rattachement

Les deux groupes métier info et tech sont créés avec addgroup. Le rattachement se fait ensuite avec usermod -aG (append to Group) qui ajoute l'utilisateur au groupe sans effacer ses autres appartenances.

```
# Création des deux groupes métier
sudo addgroup tech
sudo addgroup info

# Rattachement : badg1-3 dans tech, badg4-6 dans info
sudo usermod -aG tech badg1
sudo usermod -aG tech badg2
sudo usermod -aG tech badg3
sudo usermod -aG info badg4
sudo usermod -aG info badg5
sudo usermod -aG info badg6
```

> **ATTENTION AU FLAG -A**
>
> usermod -G tech badg1 remplacerait tous les groupes secondaires de badg1 par uniquement tech. Avec -aG (append), on ajoute tech aux groupes déjà présents. Cette erreur peut déconnecter un utilisateur de services critiques (sudo, docker, etc.) sans s'en rendre compte immédiatement : c'est l'un des pièges les plus fréquents en administration Linux.



Création des groupes info et tech, puis rattachement des utilisateurs

4 ADMINISTRATION À DISTANCE VIA SSH DURCI

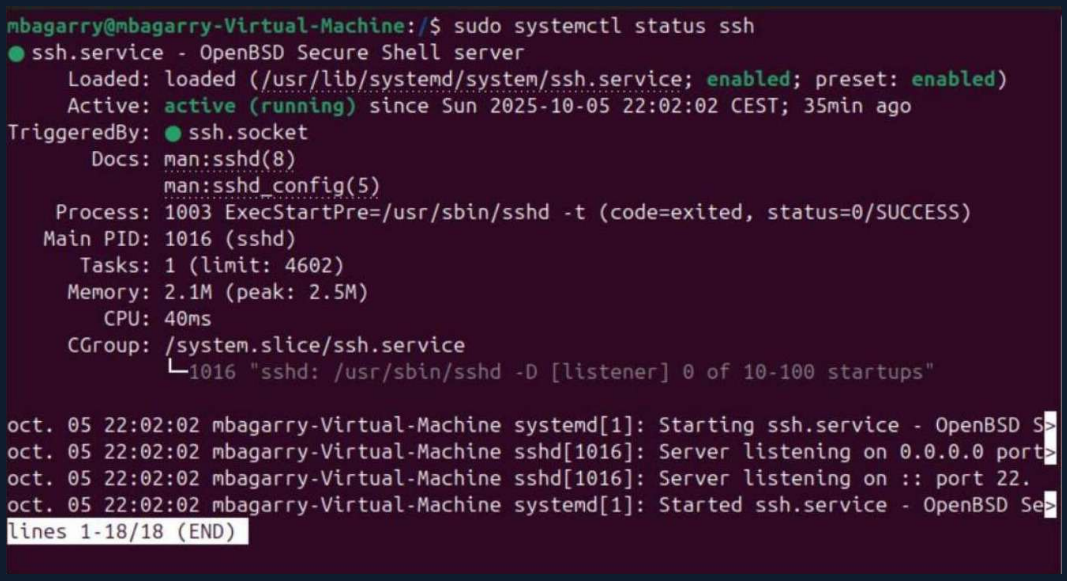
Le système de badges est exploité par une société extérieure qui intervient à distance. Un serveur SSH est donc indispensable. Par défaut, OpenSSH autorise la connexion de root, ce qui constitue la première cible des attaques automatisées : tout compte root compromis donne l'accès intégral au système. La configuration est modifiée pour interdire cette connexion.

4.1 Installation et activation du service

```
# Installation (paquet openssh-server)
sudo apt-get install openssh-server -y

# Vérification de l'état du service
sudo systemctl status ssh

# Activation au démarrage et lancement si besoin
sudo systemctl enable ssh
sudo systemctl start ssh
```



Service SSH actif et enabled dès le démarrage

4.2 Interdiction de la connexion root

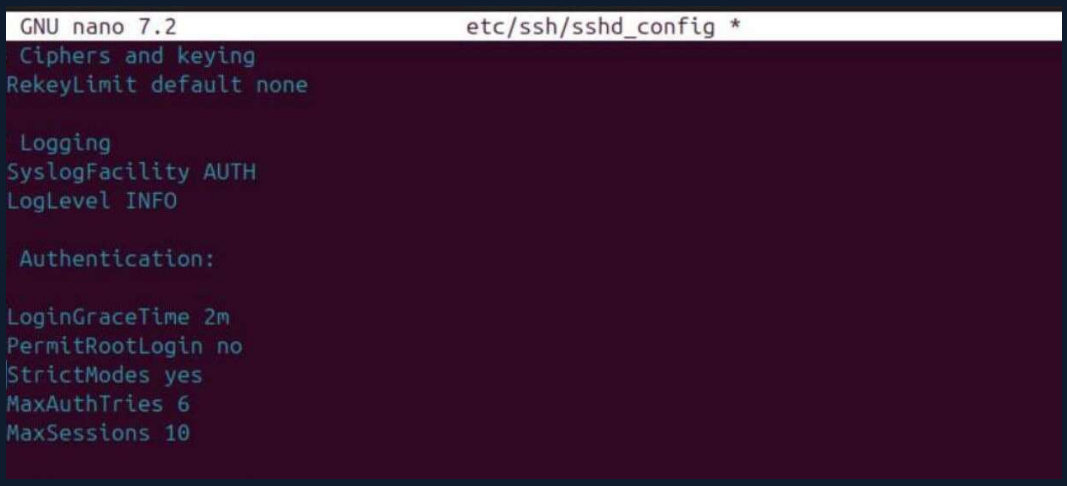
La directive PermitRootLogin dans /etc/ssh/sshd_config contrôle l'autorisation. Sa valeur par défaut (prohibit-password sur Ubuntu récent) autorise encore root par clé. Elle est passée explicitement à no pour fermer totalement la porte, forçant une connexion nominative suivie d'un sudo si une action privilégiée est nécessaire.

```
# Édition du fichier de configuration
sudo nano /etc/ssh/sshd_config

# Directive à décommenter et positionner à "no"
PermitRootLogin no
```

▸ **POURQUOI INTERDIRE LA CONNEXION ROOT**

Le compte root existe sur toutes les distributions Linux avec un login identique : c'est la cible statistique numéro un des attaques par force brute, qui peuvent se concentrer sur un seul nom d'utilisateur. Interdire sa connexion SSH élimine d'un coup cette surface. Toute action privilégiée passe alors obligatoirement par un compte nominatif puis sudo, ce qui laisse une trace dans /var/log/auth.log et permet l'imputabilité.



Directive PermitRootLogin no dans /etc/ssh/sshd_config

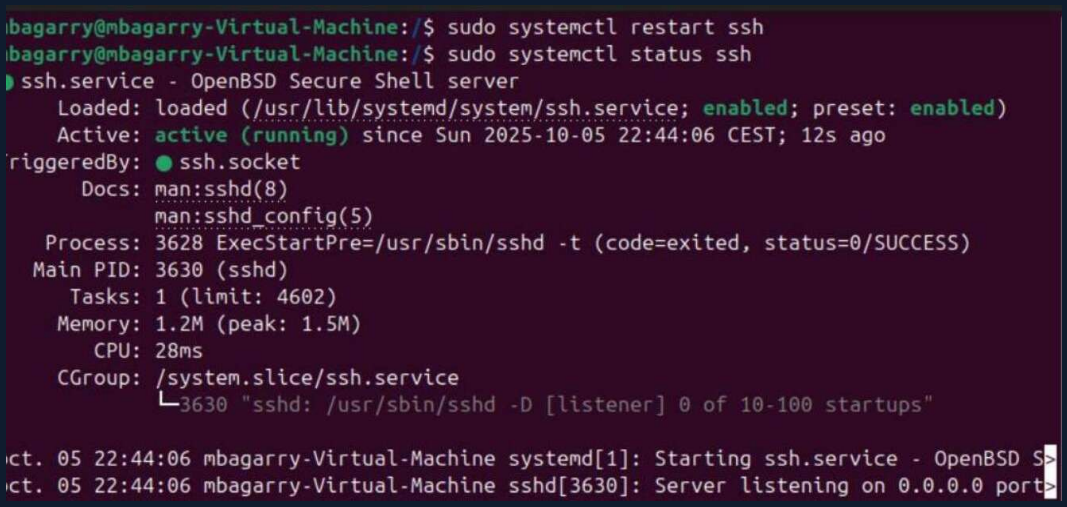
4.3 Rechargement du service et test

Toute modification de sshd_config exige un redémarrage du service pour être prise en compte. L'enchaînement restart puis status vérifie que le service a bien redémarré sans erreur — une erreur de syntaxe dans le fichier empêcherait le redémarrage et laisserait un SSH potentiellement inaccessible.

```
sudo systemctl restart ssh
sudo systemctl status ssh
```

▸ BONNE PRATIQUE OPÉRATIONNELLE

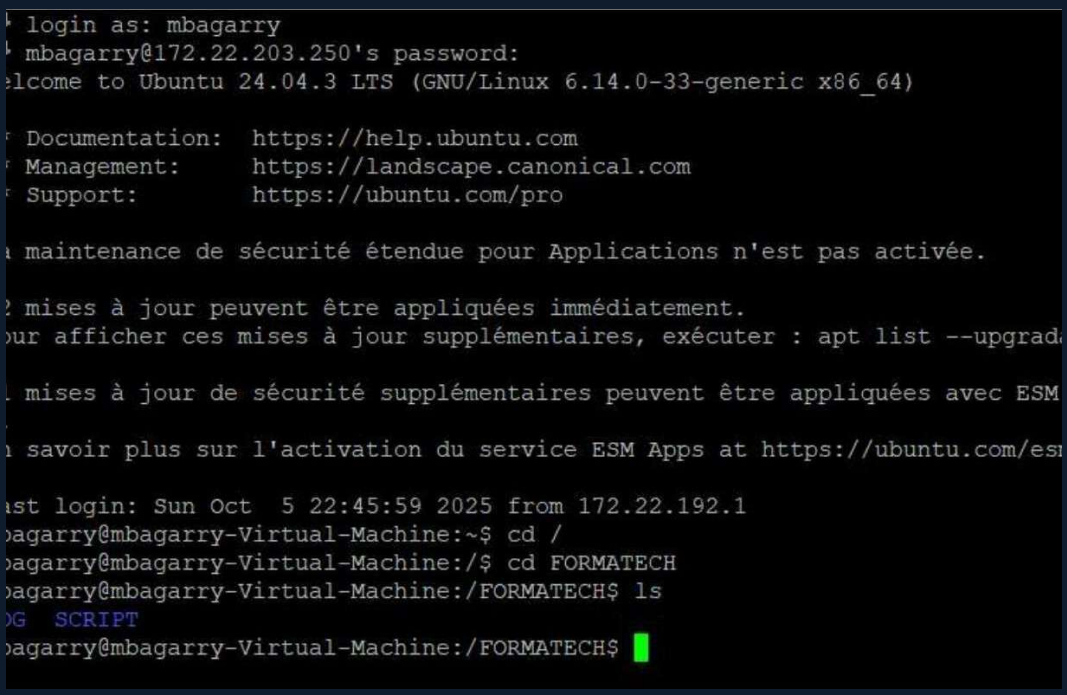
Avant de redémarrer SSH sur un serveur distant, garder une session SSH déjà ouverte et active. Si le redémarrage casse quelque chose, la session existante reste fonctionnelle pour corriger. La commande sudo sshd -t valide aussi la syntaxe sans toucher au service : c'est ce qu'ExecStartPre lance automatiquement (visible dans systemctl status).



Redémarrage réussi : nouveau PID, service actif

4.4 Connexion depuis un client Windows

L'adresse IP de la VM est récupérée avec ip a, puis la connexion est testée depuis l'hôte Windows avec PuTTY ou MobaXterm. La réception de la bannière Ubuntu et du prompt utilisateur valide le fonctionnement du service.



Connexion SSH réussie depuis PuTTY vers la VM Ubuntu

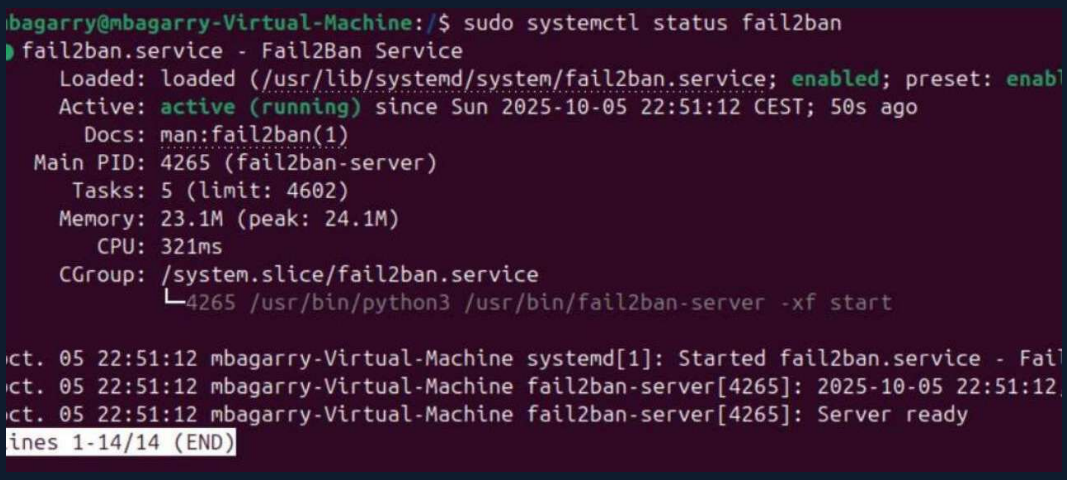
5 PROTECTION ANTI-FORCE-BRUTE AVEC FAIL2BAN

Même avec la connexion root interdite, un attaquant peut tenter des combinaisons login/mot de passe sur les comptes nominatifs. Fail2Ban surveille les journaux d'authentification, détecte les échecs répétés depuis une même IP et bannit l'adresse par ajout automatique d'une règle iptables. C'est le complément indispensable à un SSH exposé.

5.1 Installation et activation

```
# Installation silencieuse (-y accepte toutes les demandes)
sudo apt-get install fail2ban -y

# Vérification et activation au démarrage
sudo systemctl status fail2ban
sudo systemctl enable fail2ban
sudo systemctl start fail2ban
```



Service Fail2Ban actif, daemon fail2ban-server en écoute

5.2 Paramétrage du bannissement

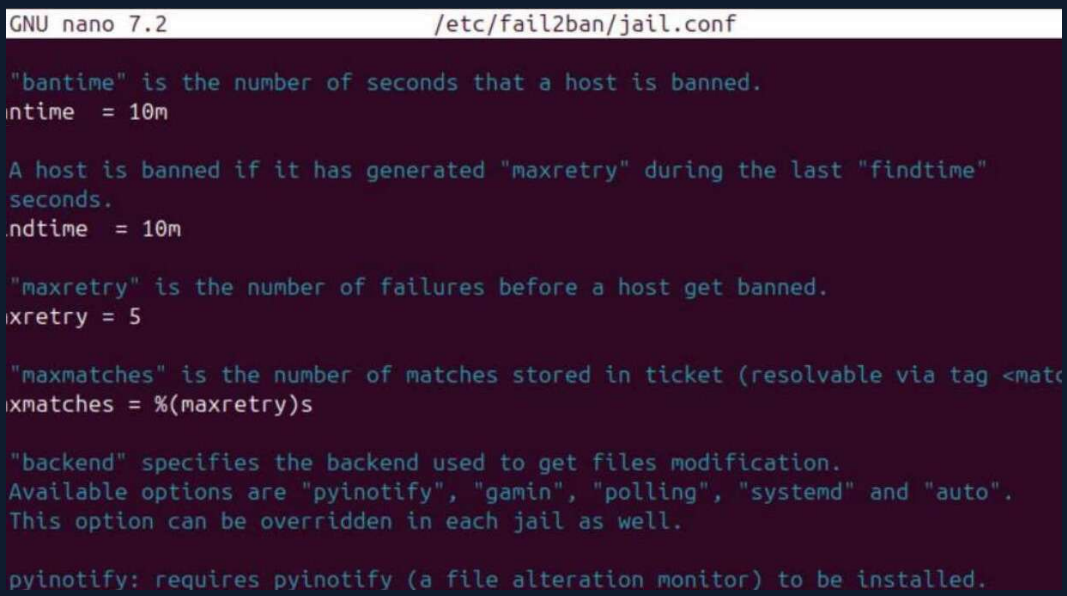
Le fichier /etc/fail2ban/jail.conf contient la configuration par défaut. Les trois paramètres clés sont bantime (durée du bannissement), findtime (fenêtre de détection) et maxretry (nombre d'échecs tolérés). La configuration par défaut convient à la cible : 5 échecs dans une fenêtre de 10 minutes déclenchent un bannissement de 10 minutes.

```
sudo nano /etc/fail2ban/jail.conf

# Paramètres par défaut :
bantime  = 10m    # Durée du bannissement
findtime = 10m    # Fenêtre de détection des échecs
maxretry = 5      # Nombre d'échecs avant bannissement
```

▸ **BONNE PRATIQUE : UTILISER JAIL.LOCAL**

En production, jail.conf n'est pas modifié directement car il est écrasé à chaque mise à jour du paquet fail2ban. Les sections utiles sont copiées dans /etc/fail2ban/jail.local, lu en dernier et prévalant sur jail.conf. Dans le cadre de ce projet, l'édition directe de jail.conf a été conservée pour simplifier la démonstration, mais la commande recommandée en production serait sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local.



Extrait de jail.conf : bantime, findtime et maxretry=5 bien positionnés

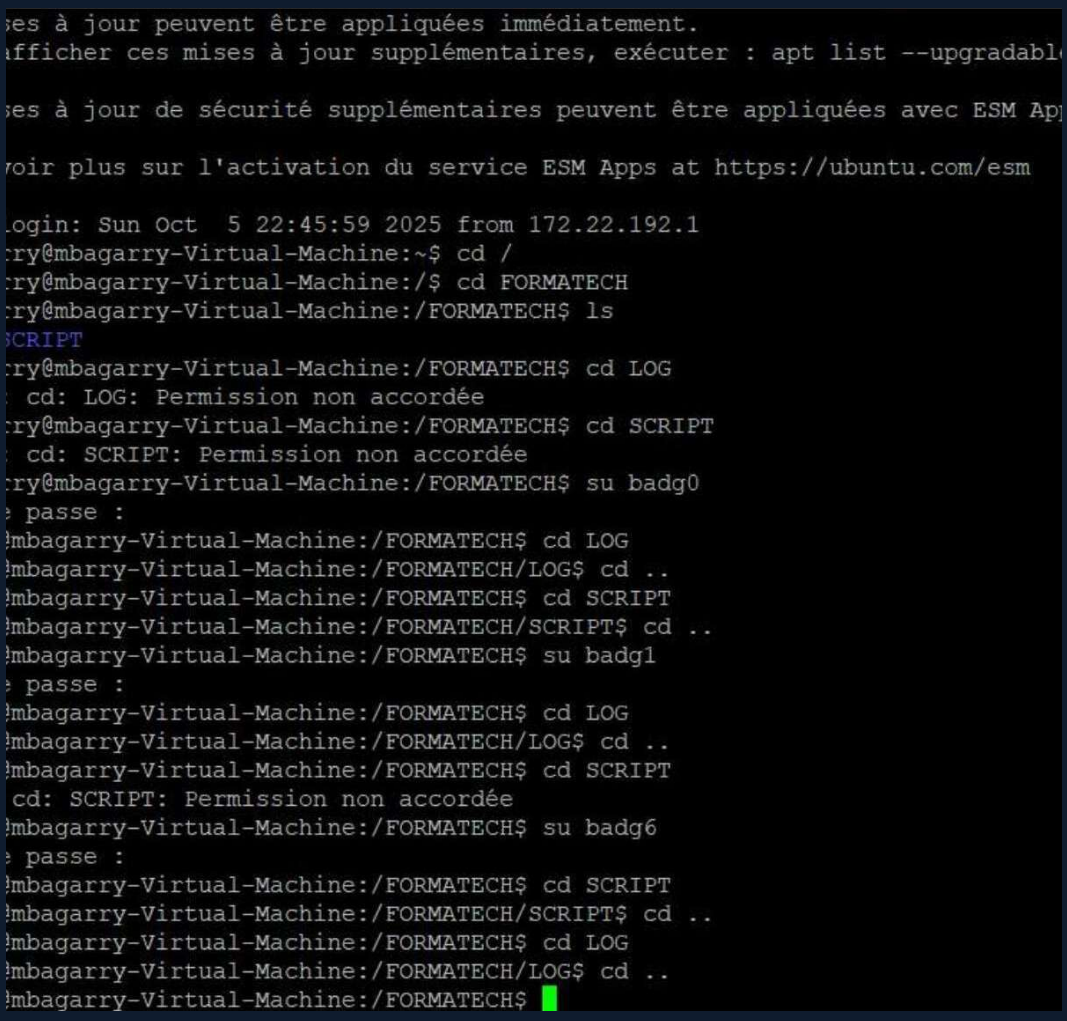
6 RECETTE ET VALIDATION TECHNIQUE

La phase de recette valide trois aspects distincts : la correction des permissions (chaque utilisateur accède uniquement à ce qui lui est autorisé), l'efficacité de Fail2Ban (l'IP attaquante est effectivement bannie après 5 tentatives) et la traçabilité (la liste des IP bannies est consultable).

6.1 Test de la matrice de droits

Le test consiste à se connecter en SSH depuis le poste Windows, puis à basculer successivement sur chaque profil utilisateur (su badg0, su badg1, su badg6) et à tenter d'entrer dans chaque dossier. Le résultat attendu doit coïncider exactement avec la matrice définie en section 1.2.

Compte	cd LOG	cd SCRIPT	Résultat attendu
`mbagarry (admin)`	Refusé	Refusé	Conforme — ni info ni tech
`badg0 (maître)`	Autorisé	Autorisé	Conforme — propriétaire
`badg1 (tech)`	Autorisé	Refusé	Conforme — tech sans SCRIPT
`badg6 (info)`	Autorisé	Autorisé	Conforme — info a les deux



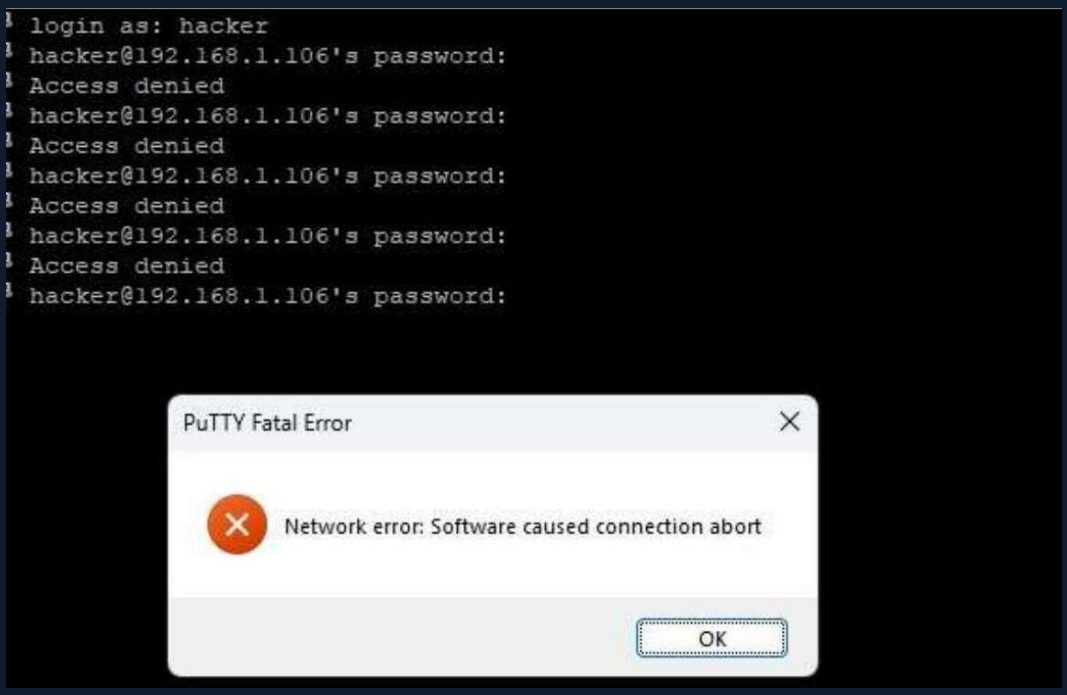
Session PuTTY : bascule entre les profils et vérification des accès par dossier

6.2 Test anti-force-brute Fail2Ban

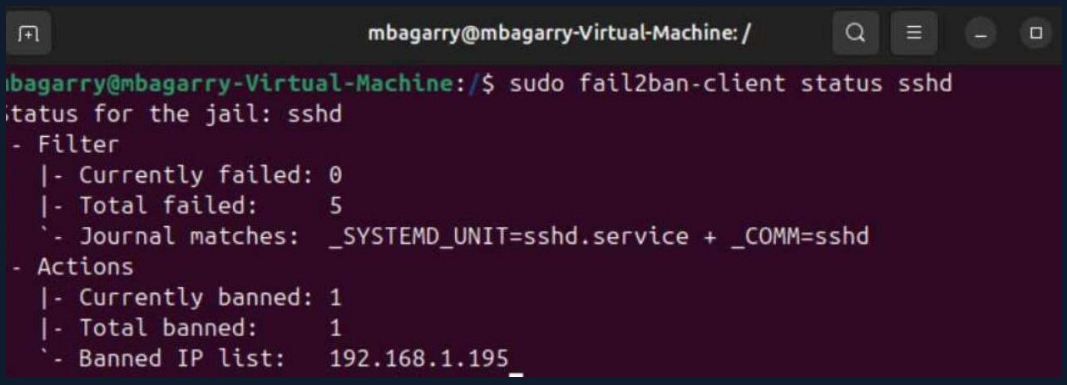
Depuis un second poste (IP 192.168.1.195), un pseudo-attaquant « hacker » enchaîne des tentatives de connexion SSH avec des mots de passe invalides. Après 5 échecs dans la fenêtre de détection, Fail2Ban bascule l'IP source en règle de blocage iptables. La connexion est brutalement coupée côté client (message « Network error: Software caused connection abort »).

▸ **CRITÈRE DE SUCCÈS**

Côté attaquant : rupture de connexion PuTTY avec erreur réseau après 5 essais. Toute nouvelle tentative depuis la même IP est refusée pendant la durée du bantime (10 minutes par défaut). Côté serveur : la commande sudo fail2ban-client status sshd affiche Currently banned: 1 et liste l'IP dans Banned IP list.



Côté attaquant : après 5 échecs, PuTTY perd la connexion (erreur fatale réseau)



Côté serveur : fail2ban-client status sshd confirme le bannissement de 192.168.1.195

6.3 Commandes de diagnostic et exploitation

Commande	Usage
`sudo fail2ban-client status`	Liste des jails actives
`sudo fail2ban-client status sshd`	Détail de la jail SSH : IPs bannies, compteurs
`sudo fail2ban-client unban <IP>`	Débannissement manuel (faux positif)
`sudo tail -f /var/log/fail2ban.log`	Suivi en temps réel des bans et débans
`sudo tail -f /var/log/auth.log`	Journal des tentatives SSH (échecs et succès)
`getfacl /FORMATECH/LOG`	Vérification des ACL étendues sur un dossier
`groups <utilisateur>`	Liste des groupes d'appartenance d'un compte
`id <utilisateur>`	UID, GID principal et GIDs secondaires

6.4 Bilan et axes d'amélioration

Le projet répond à l'ensemble des exigences fonctionnelles formulées : arborescence cloisonnée, droits conformes à la matrice, SSH durci, Fail2Ban opérationnel. Trois pistes d'évolution restent identifiées pour une mise en production réelle.

▸ LIMITES DE L'IMPLÉMENTATION ACTUELLE

(1) L'authentification SSH reste en mot de passe. En production, la bascule sur authentification par clé publique (PubkeyAuthentication yes, PasswordAuthentication no) supprimerait totalement la surface d'attaque par force brute. (2) Le bannissement Fail2Ban est non persistant : un redémarrage du serveur vide la liste. L'ajout de action = %(action_mwl)s et d'une base persistante (fail2ban.sqlite3) serait nécessaire pour conserver les bans. (3) Aucune supervision centralisée : les logs restent locaux. Une remontée syslog vers un puits centralisé (rsyslog distant, SIEM type Wazuh) serait la suite logique pour une infrastructure réelle.