MATHÉO PINGET

FISA INFO 24/27

MÉMOIRE TECHNIQUE

SYSTÈME DE RÉCUPÉRATION ET SUPPRESSION SÉCURISÉE DES MOTS DE PASSE VAULT

RÉSUMÉ

Ce mémoire présente la conception et la mise en œuvre d'un système sécurisé et automatisé de gestion des mots de passe d'infrastructure via HashiCorp Vault, Ansible et GitLab CI/CD.

Le projet vise à supprimer les manipulations manuelles sources d'erreurs ou de fuites de données, et à répondre aux exigences de sécurité, traçabilité et conformité dans un contexte DevOps.

Le système permet de récupérer ou supprimer des secrets en toute sécurité, avec journalisation, chiffrement et contrôle d'accès strict.

SOMMAIRE

INTRODUCTION

1. Introduction

Dans un contexte de transformation numérique accélérée, la gestion sécurisée des mots de passe d'infrastructure représente un enjeu stratégique majeur pour les entreprises. Au sein d'Orange Business, cette gestion suit encore des processus largement manuels, chronophages et sources de risques sécuritaires.

Ce mémoire technique présente la conception et la mise en œuvre d'un système automatisé et sécurisé de gestion des mots de passe d'infrastructure, s'appuyant sur HashiCorp Vault, Ansible et GitLab CI/CD.

L'objectif principal est double : éliminer les risques inhérents aux manipulations manuelles (erreurs humaines, temps de traitement, exposition des données sensibles) et répondre aux exigences de sécurité, de traçabilité et de conformité.

La solution développée intègre chiffrement des données, contrôle d'accès granulaire, journalisation complète des opérations et automatisation des processus de récupération et de suppression des secrets. Elle s'inscrit dans une démarche DevSecOps, plaçant la sécurité au cœur du cycle de développement et d'exploitation.

Ce mémoire détaille la démarche adoptée, du diagnostic initial à la mise en œuvre technique, en passant par la conception de la solution et son intégration dans l'environnement existant.

PRÉSENTATION DE L'ENTREPRISE

2.1 Historique

Orange Business (anciennement Orange Business Services) est une filiale et une marque commerciale du groupe Orange, spécialisée dans la fourniture de services de communication intégrée aux entreprises, tant en France qu'à l'international. Les domaines d'expertise d'Orange Business incluent le cloud computing, les télécommunications, les communications unifiées et la collaboration. Cette entité regroupe les activités business to business des différentes filiales françaises et internationales du groupe Orange.

Créée le 1er juin 2006 sous le nom d'Orange Business Services, la marque a été constituée pour offrir une identité unique aux services de télécommunication (données, téléphonie et convergence) et informatiques destinés aux entreprises. Orange Business opère dans 65 pays. La consolidation et le regroupement sous une seule marque ont permis de réunir des entités issues de trois entreprises aujourd'hui dissoutes : France Telecom, Equant et Wanadoo.

Le 16 février 2023, Christel Heydemann, directrice générale du groupe Orange, a annoncé que la marque Orange Business Services serait désormais simplifiée en Orange Business.

2.2 Secteur d'activité

- **Description générale** : OBS opère dans le secteur des services digitaux, offrant des solutions de communication, de cloud computing, de cybersécurité, et de gestion des données.

- **Marché cible** : OBS cible principalement les grandes entreprises et multinationales, offrant des solutions adaptées à leurs besoins complexes.

PRÉSENTATION DE L'ENTREPRISE

2.3 Produits et Services

Orange Business est organisé en plusieurs divisions et sous-divisions, chacune dédiée à des missions spécifiques pour répondre aux besoins variés de ses clients.

Je travaille dans l'entité Operations, qui relève de la structure plus large de Global Delivery & Operations (GDO). Le rôle de GDO est de gérer les opérations à l'échelle mondiale, garantissant ainsi la fiabilité et la performance des services fournis aux clients d'Orange Business.

Au sein de GDO, j'appartiens plus précisément au service Global Platforms & Services (GPS). Ce service est crucial pour Orange Business, car il est responsable de la création de valeur dans des domaines stratégiques tels que le Cloud et la Cyberdéfense.

GPS soutient l'innovation en développant des solutions techniques de pointe, en fournissant des expertises avancées, et en appliquant des processus d'industrialisation rigoureux.

En combinant ces éléments, GPS contribue activement à accélérer la croissance des propositions de valeur stratégiques (SVP) d'Orange Business. Ces SVP sont conçues pour renforcer la compétitivité d'Orange Business sur le marché et offrir des solutions qui répondent aux enjeux actuels de transformation numérique des entreprises clientes.

PRÉSENTATION DU SUJET ET DES OBJECTIFS

3.1 Contexte

Dans le cadre de la gestion des infrastructures virtuelles, il est souvent nécessaire de transmettre des mots de passe d'administration à d'autres équipes, notamment pour les renseigner dans leurs propres outils. Historiquement, cette transmission était réalisée manuellement : les mots de passe étaient extraits un par un, puis envoyés de manière sécurisée, souvent par email chiffré. Ce processus devient rapidement chronophage et source d'erreurs lorsqu'il s'agit de gérer un parc important de machines virtuelles.

Par exemple, pour transmettre les accès de 40 machines, il fallait récupérer individuellement chaque mot de passe, les consigner dans un fichier, le chiffrer, puis l'envoyer — ce qui mobilise un temps considérable et augmente les risques liés à la sécurité (erreurs, oublis, mauvaise manipulation).

3.2 Enjeux et objectifs

Ce projet a pour but de concevoir une solution permettant de :

- Sécuriser la transmission des mots de passe à une autre équipe sans risque de fuite ;
- Automatiser la récupération et la mise en forme des mots de passe pour limiter l'intervention humaine ;
- Fiabiliser les échanges grâce à un chiffrement systématique et l'utilisation de solutions centralisées comme Vault ;
- Gagner du temps dans les opérations de masse (récupération ou suppression de comptes utilisateurs) ;
- Faciliter l'intégration avec les outils DevOps existants (Ansible, GitLab CI/CD) pour un usage fluide et automatisé.

PRÉSENTATION DU SUJET ET DES OBJECTIFS

3.3 État des lieux

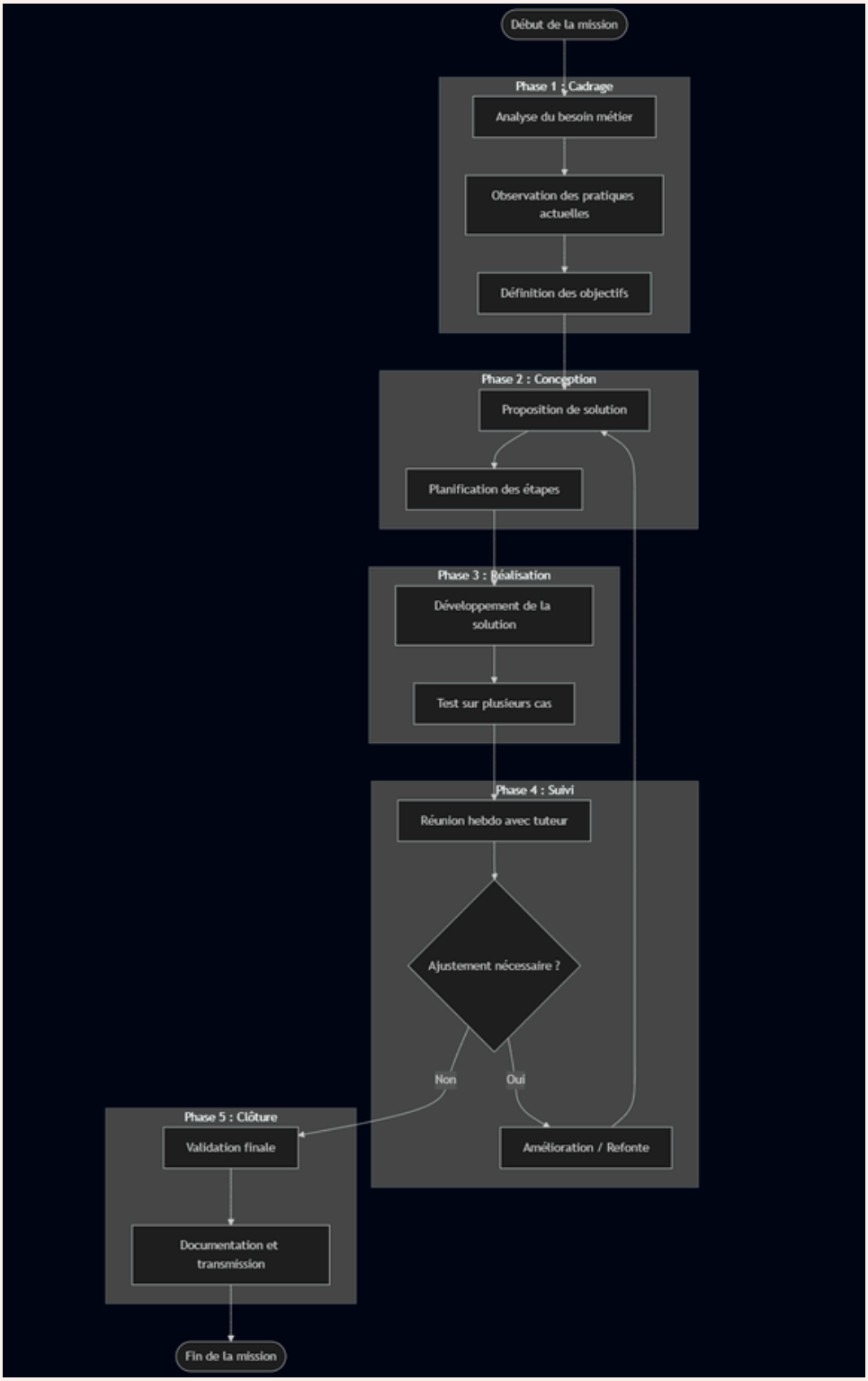
Avant la mise en place de cette solution :

- La récupération des mots de passe se faisait manuellement dans le Vault, machine par machine ;
- Les fichiers générés étaient transmis par email chiffré, mais sans standardisation ni automatisation ;
- Aucun système centralisé ne permettait de stocker et partager la clé de chiffrement de manière sécurisée ;
- L'ensemble du processus n'était pas intégré dans les chaînes CI/CD ni dans les outils d'automatisation comme Ansible ;
- Les suppressions d'utilisateurs, lorsqu'elles devaient être faites en masse, étaient elles aussi manuelles, longues et sujettes à erreurs.

La solution développée permet désormais de :

- Récupérer automatiquement les mots de passe d'un ou plusieurs utilisateurs sur un ou plusieurs hôtes (ou groupes Ansible) ;
- Générer un fichier CSV formaté contenant les identifiants ;
- Chiffrer ce fichier dans une archive ZIP protégée ;
- Stocker la clé de chiffrement dans un Vault sécurisé ;
- Permettre à l'équipe destinataire de récupérer la clé de manière autonome dans Vault et d'ouvrir l'archive ;
- Automatiser également les opérations de suppression de comptes utilisateurs, toujours en s'appuyant sur Ansible.

LOGIGRAMME DES PRINCIPALES ÉTAPES DE LA MISSION



DÉVELOPPEMENT DE LA MISSION

5.1 Analyse de l'existant et des besoins

5.1.1 Analyse critique de l'existant

L'analyse approfondie de l'existant a révélé plusieurs problématiques majeures dans la gestion des mots de passe d'infrastructure :

Processus manuel actuel :

- Récupération individuelle des mots de passe depuis HashiCorp Vault via interface web
- Compilation manuelle dans des fichiers non standardisés
- Transmission par email chiffré sans centralisation de la clé
- Aucune traçabilité des accès aux mots de passe transmis
- Temps de traitement : 2-3 minutes par machine (soit 2h pour 40 machines)

Risques identifiés :

- Sécurité : Exposition des mots de passe durant la compilation manuelle
- Erreur humaine : Risque de copier-coller incorrect ou d'omission
- Scalabilité : Processus non viable pour des volumes importants
- Conformité : Absence de logs d'audit et de traçabilité

Impact quantifié :

- 40 machines = 2h de travail manuel
- Coût opérationnel mensuel : 8-10h pour l'équipe

DÉVELOPPEMENT DE LA MISSION

5.1 Analyse de l'existant et des besoins

5.1.2 Expression des besoins

Besoins fonctionnels :

a. Automatisation : Récupération automatisée des mots de passe
b. Sécurisation : Chiffrement fort et gestion sécurisée des clés
c. Scalabilité : Support de 1 à n machines simultanément
d. Intégration : Compatibilité avec l'écosystème DevOps existant
e. Traçabilité : Logging complet des opérations

Besoins non-fonctionnels :

- Performance : Traitement de 100 machines en <5 minutes
- Fiabilité : Disponibilité 99.9%
- Sécurité : Chiffrement AES-256
- Maintenabilité : Architecture modulaire et documentée

5.2 Recherche documentaire et choix techniques

5.2.1 Recherche bibliographique(**Annexe 8**)

Sources consultées :

- Documentation officielle HashiCorp Vault (API v1)
- Best practices DevSecOps de l'OWASP
- Standards de sécurité ISO 27001/27002
- Documentation Ansible Community Collection
- Guides GitLab CI/CD Enterprise

Entretiens réalisés :

- Équipe destinataire : Besoins d'utilisation finale

5.2.2 Architecture technique retenue

Stack technologique :

- Orchestration : GitLab CI/CD
- Automatisation : Ansible Core 2.12+
- Sécurité : HashiCorp Vault, OpenSSL

Patterns architecturaux appliqués :

- Separation of Concerns : Séparation récupération/suppression
- Fail-Fast : Validation précoce des paramètres
- Defense in Depth : Multiples couches de sécurité

5.3 Conception de la solution

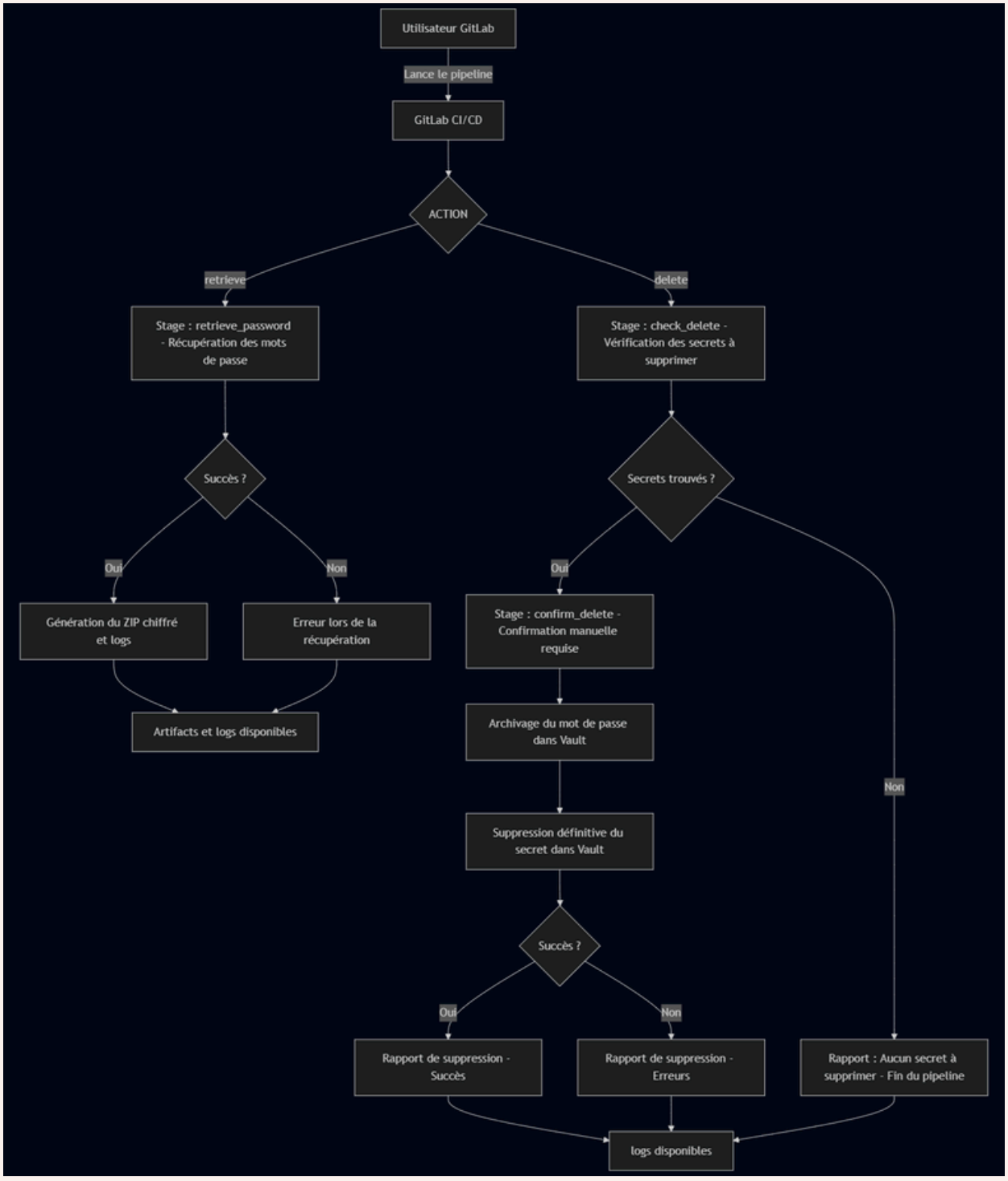
5.3.1 Architecture globale

Le diagramme d'architecture illustre le flux complet de traitement du pipeline GitLab CI/CD pour la gestion des mots de passe.

L'architecture suit un modèle de workflow conditionnel basé sur l'action demandée.

DÉVELOPPEMENT DE LA MISSION

LE DIAGRAMME D'ARCHITECTURE



5.3.1 Architecture globale

Flux principal d'exécution

Le pipeline s'articule autour d'un point de décision central basé sur la variable **ACTION** qui détermine le comportement du système :

Branche "retrieve" (Récupération)

Lorsque l'action est définie sur "retrieve", le pipeline suit le chemin suivant :

- **Stage retrieve_password** : Récupération des mots de passe depuis Vault
- **Point de décision "Succès ?"** : Validation du succès de l'opération
- **En cas de succès** : Génération du ZIP chiffré et des logs
- **En cas d'échec** : Erreur lors de la récupération
- **Finalisation** : Artifacts et logs disponibles

Branche "delete" (Suppression)

Lorsque l'action est définie sur "delete", le pipeline suit un processus de validation en deux étapes :

- **Stage check_delete** : Vérification des secrets à supprimer
- **Point de décision "Secrets trouvés ?"** : Validation de l'existence des secrets
- **Si secrets trouvés** : Passage au stage de confirmation manuelle
- **Stage confirm_delete** : Confirmation manuelle requise
- **Archivage du mot de passe** : Sauvegarde avant suppression
- **Suppression définitive** : Effacement du secret dans Vault
- **Génération des rapports** : Logs de traçabilité

5.3.1 Architecture globale

Mécanismes de sécurité intégrés

L'architecture intègre plusieurs points de contrôle :

- **Validation manuelle** : Les opérations critiques nécessitent une intervention humaine
- **Vérification préalable** : Contrôle de l'existence des secrets avant suppression
- **Archivage** : Sauvegarde des mots de passe avant suppression
- **Traçabilité** : Génération systématique de logs pour audit

5.3.2 Conception détaillée du rôle Ansible

Le rôle Ansible *vault_password_manager* (**Annexe 2**) constitue le cœur métier de la solution. Sa structure modulaire permet une gestion flexible et sécurisée des mots de passe.

Structure du rôle

Organisation des tâches

Le rôle est structuré autour de tâches spécialisées chaque tâches est en (**Annexe 3**):

Tâches principales :

- *main.yml* : Dispatcher principal et validation
- *main_vault_retrieve.yml* : Orchestration récupération
- *main_vault_delete.yml* : Orchestration suppression
- *main_vault_check_delete.yml* : Vérifications pré-suppression

5.3.2 Conception détaillée du rôle Ansible

Tâches opérationnelles :

- *retrieve_password_for_host.yml* : Récupération par hôte
- *delete_password_for_host.yml* : Suppression par hôte
- *check_users_for_host.yml* : Vérification utilisateurs
- *setup_zip_password.yml* : Gestion ZIP et sécurité
- *log_operation.yml* : Audit et traçabilité

5.3.3 Pipeline GitLab CI/CD

Vue d'ensemble

Le pipeline GitLab CI/CD développé permet la gestion sécurisée des mots de passe stockés dans HashiCorp Vault. Il s'articule autour de trois actions principales : récupération, vérification et suppression des mots de passe pour une liste d'hôtes définie. Code entier gitlab-ci.yml(**Annexe 4**)

Architecture du pipeline

Le pipeline est structuré en trois étapes principales :

a. **retrieve_password** : Récupération sécurisée des mots de passe
b. **check_delete** : Vérification des éléments avant suppression
c. **confirm_delete** : Confirmation et exécution de la suppression

5.3.3 Pipeline GitLab CI/CD

Jobs du pipeline

Job de récupération (retrieve_vm_passwords)

Objectif : Récupérer de manière sécurisée les mots de passe stockés dans Vault

Fonctionnement :

- Exécution manuelle uniquement (when: manual)
- Génération d'un mot de passe ZIP aléatoire pour sécuriser le zip
- Appel du playbook Ansible avec l'action "retrieve"
- Génération d'artifacts chiffrés (passwords.zip + logs)

Job de vérification (check_vm_password)

Objectif : Vérifier les éléments avant suppression

Fonctionnement :

- Activation automatique si ACTION=delete
- Exécution du playbook avec l'action "check_delete"
- Validation des droits et existence des mots de passe

Job de confirmation (confirm_vm_password)

Objectif : Confirmer et exécuter la suppression

Fonctionnement :

- Exécution manuelle pour validation humaine
- Appel du playbook avec l'action "delete"
- Génération de logs de traçabilité

5.4 Réalisation

5.4.1 Mise en place de l'environnement

Préparation de l'environnement de développement :

Pour ce projet, j'ai utilisé mon environnement WSL (Windows Subsystem for Linux) déjà configuré dans Visual Studio Code. Cela m'a permis de bénéficier d'un environnement de développement intégré tout en utilisant les outils Linux. J'ai adapté cet environnement pour répondre aux besoins spécifiques du projet :

- Configuration d'Ansible : J'ai configuré Ansible sur mon environnement WSL pour l'automatisation des tâches.
- Accès à GitLab d'entreprise : J'ai configuré l'accès au GitLab d'entreprise existant pour gérer le code source et les pipelines CI/CD.

Configuration des accès et permissions :

Étant donné que GitLab et Vault étaient déjà existants, j'ai configuré les accès et permissions nécessaires pour interagir avec ces systèmes :

- Création d'un nouveau projet GitLab : J'ai créé un nouveau projet GitLab pour ce projet spécifique et configuré les accès nécessaires.
- Configuration des accès sécurisés : J'ai configuré les accès sécurisés entre mon environnement et GitLab pour garantir que les communications étaient chiffrées et sécurisées.

5.4.2 Développement du rôle Ansible

Création du rôle Ansible :

J'ai développé un rôle Ansible unique pour automatiser à la fois la récupération et la suppression des mots de passe. Ce rôle est conçu pour être exécuté par le runner CI de GitLab. Voici comment j'ai structuré ce rôle :

- **Fonctionnalités du rôle :** Le rôle utilise des modules Ansible pour récupérer les secrets nécessaires depuis Vault, les stocker de manière sécurisée, vérifier leur existence, les archiver si nécessaire, et les supprimer de manière sécurisée.
- **Modularité :** Bien que ce soit un seul rôle, j'ai veillé à ce qu'il soit modulaire, avec des tâches distinctes pour la récupération et la suppression, afin de faciliter la maintenance et les mises à jour.

Intégration avec HashiCorp Vault :

L'intégration avec Vault a été gérée par le runner CI de GitLab. Le runner CI est configuré pour interagir directement avec Vault, ce qui simplifie la récupération et la suppression des secrets.

J'ai mis en place des mécanismes de gestion des erreurs pour garantir la robustesse des opérations et configuré des logs détaillés pour la traçabilité.

5.4.3 Automatisation avec GitLab CI/CD

Création du pipeline CI/CD :

J'ai créé un pipeline CI/CD dans le projet GitLab pour automatiser les processus de récupération et de suppression des mots de passe. Voici les étapes principales du pipeline :

- **Authentification** : Configuration des accès sécurisés pour l'authentification avec Vault et la configuration SSH. Les détails de cette configuration sont disponibles en annexe.
- **Déploiement** : Le pipeline inclut une étape de déploiement qui installe les outils et dépendances nécessaires, configure Vault et Ansible, et prépare l'environnement pour l'exécution des tâches.
- **Étape de récupération** : Utilisation du rôle Ansible pour extraire les secrets de Vault et les stocker de manière sécurisée.
- **Étape de vérification** : Vérification de l'existence des secrets avant suppression et archivage des mots de passe.
- **Étape de suppression** : Utilisation du rôle Ansible pour supprimer les secrets de Vault.

Gestion des artefacts et des logs :

J'ai configuré la gestion des artefacts et des logs pour assurer la traçabilité des opérations. Les logs détaillés pour chaque étape du pipeline exécutée par le runner CI sont générés et stockés de manière sécurisée pour consultation ultérieure.

5.4.4 Tests et validation

Stratégie de test :

J'ai mis en place une stratégie de test pour valider le bon fonctionnement du système :

- Tests de bout en bout : Validation du fonctionnement global du système, de la récupération à la suppression des secrets, exécutée par le runner CI.

5.4.5 Documentation

Création de la documentation (Annexe 5) :

J'ai créé une documentation complète pour le projet, incluant des guides d'utilisation pour les utilisateurs finaux, des manuels techniques pour les administrateurs, et des notes de version pour documenter les changements et améliorations.

5.5 Difficultés rencontrées et solutions apportées

Difficultés :

- **Automatisation des tâches** : Automatiser les tâches de récupération et de suppression des mots de passe tout en garantissant la sécurité et la traçabilité a été complexe.
- **Gestion des erreurs** : La mise en place d'un système robuste de gestion des erreurs pour garantir la fiabilité des opérations a nécessité plusieurs itérations.

Solutions :

- **Utilisation de rôles Ansible modulaires** : J'ai développé un rôle Ansible modulaire qui a permis de séparer les tâches de récupération et de suppression, facilitant ainsi la maintenance et les mises à jour.
- **Logs détaillés et tests rigoureux** : J'ai mis en place des logs détaillés et effectué des tests rigoureux pour identifier et résoudre les problèmes de gestion des erreurs.

BILAN

Résultats obtenus :

- **Automatisation réussie** : Le système permet désormais une automatisation complète de la récupération et de la suppression des mots de passe, réduisant ainsi les erreurs humaines et le temps de traitement.
- **Sécurité renforcée** : L'intégration avec HashiCorp Vault et l'utilisation de logs détaillés ont permis de renforcer la sécurité et la traçabilité des opérations.
- **Efficacité opérationnelle** : Le temps nécessaire pour gérer les mots de passe a été considérablement réduit, améliorant ainsi l'efficacité opérationnelle.

Impact sur l'entreprise :

- **Réduction des coûts** : L'automatisation a permis de réduire les coûts opérationnels en minimisant le temps consacré à la gestion manuelle des mots de passe.
- **Amélioration de la sécurité** : La sécurité des données a été améliorée grâce à l'utilisation de Vault et à la mise en place de mécanismes de gestion des erreurs robustes.
- **Satisfaction des utilisateurs** : Les utilisateurs finaux ont exprimé leur satisfaction quant à la facilité d'utilisation et à la fiabilité du nouveau système.

CONCLUSION

Résumé général :

Ce projet a permis de développer un système sécurisé et automatisé de gestion des mots de passe d'infrastructure, répondant ainsi aux besoins de sécurité, de traçabilité et d'efficacité opérationnelle. L'intégration avec HashiCorp Vault et l'utilisation de GitLab CI/CD ont été des éléments clés de la réussite de ce projet.

Réflexion personnelle :

Ce projet a été une expérience enrichissante qui m'a permis de développer mes compétences en automatisation, en sécurité des données et en gestion de projet. J'ai appris l'importance de la collaboration avec les différentes parties prenantes et de la rigueur dans la gestion des erreurs et des tests.

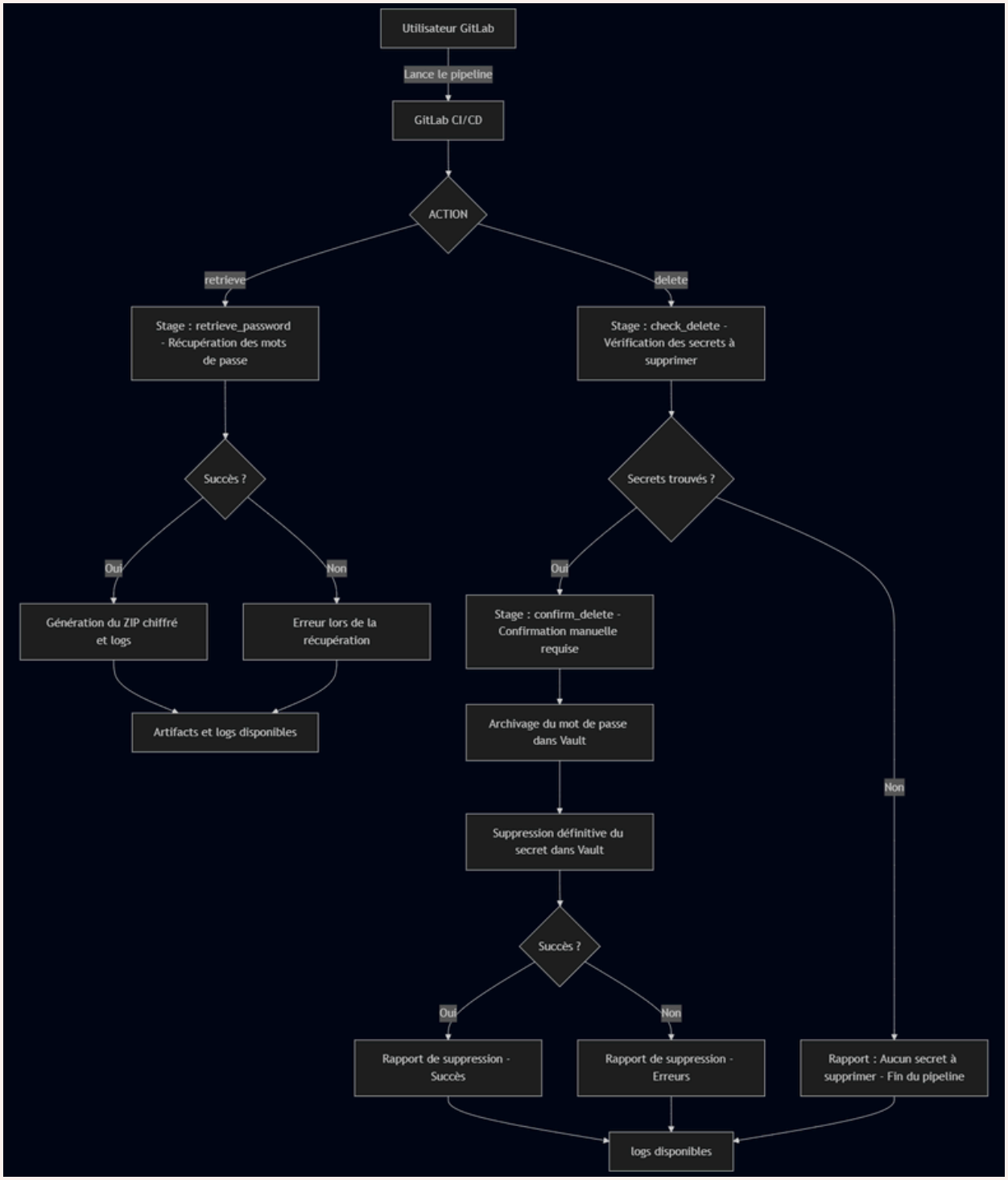
Perspectives d'avenir :

À l'avenir, il serait pertinent d'envisager des améliorations supplémentaires, telles que l'ajout de mots de passe en masse et l'intégration automatique des mots de passe récupérés dans les outils utilisés par les autres équipes, afin de limiter l'intervention humaine.

Ce projet a posé des bases solides pour une gestion des mots de passe à la fois sécurisée et automatisée, ouvrant ainsi la voie à de futures innovations dans ce domaine.

ANNEXES

Annexe 1 - Diagramme exécution de la Pipeline CI-CD



ANNEXES

Annexe 2 - arborescence rôle du ansible

```
vault_password_manager/
├── tasks/
│   ├── main.yml                      # Dispatcher principal et validation
│   ├── main_vault_retrieve.yml       # Orchestration récupération
│   ├── main_vault_delete.yml         # Orchestration suppression
│   ├── main_vault_check_delete.yml   # Vérifications pré-suppression
│   ├── retrieve_password_for_host.yml # Récupération par hôte
│   ├── delete_password_for_host.yml   # Suppression par hôte
│   ├── check_users_for_host.yml       # Vérification utilisateurs
│   ├── setup_zip_password.yml        # Gestion ZIP et sécurité
│   └── log_operation.yml             # Audit et traçabilité
├── defaults/main.yml                 # Configuration par défaut
├── vars/main.yml                     # Variables internes
├── handlers/main.yml                 # Gestionnaires d'événements
├── meta/main.yml                     # Métadonnées du rôle
├── tests/
│   ├── inventory                     # Inventaire de test
│   └── test.yml                      # Tests du rôle
└── README.md                         # Documentation
```

ANNEXES

Annexe 3

main.yml - Point d'entrée principal

```yaml
##############################################################################
# Rôle Ansible de gestion des mots de passe dans HashiCorp Vault
#
# Description:
#   Ce rôle centralise les opérations de gestion des secrets dans Vault :
#   - Récupération sécurisée des mots de passe
#   - Vérification avant suppression
#   - Suppression des secrets
#
# Paramètres:
#   action: (string) Détermine l'opération à effectuer
#     - "retrieve" : Récupération des mots de passe
#     - "delete" : Suppression des secrets
#     - "check_delete" : Vérification avant suppression
#
# Dépendances:
#   - Collection community.hashi_vault
#   - Vault configuré et accessible
#   - Token Vault avec les permissions appropriées
##############################################################################

# Tâche de récupération des mots de passe
# Activée quand action == "retrieve"
# Utilise main_vault_retrieve.yml pour le traitement détaillé
- name: Retrieve passwords from Vault
  include_tasks: main_vault_retrieve.yml
  when: action == "retrieve"

# Tâche de suppression des secrets
# Activée quand action == "delete"
# Utilise main_vault_delete.yml pour la suppression sécurisée
- name: Delete secrets from Vault
  include_tasks: main_vault_delete.yml
  when: action == "delete"

# Tâche de vérification avant suppression
# Activée quand action == "check_delete"
# Utilise main_vault_check_delete.yml pour l'analyse préalable
- name: Check secrets before deletion
  include_tasks: main_vault_check_delete.yml
  when: action == "check_delete"
```

ANNEXES

Annexe 3

main_vault_retrieve.yml - Orchestration récupération

```yaml
###############################################################################
# Playbook de récupération des mots de passe depuis HashiCorp Vault
#
# Objectif : Ce playbook permet de récupérer de manière sécurisée les mots de passe
# stockés dans HashiCorp Vault pour une liste d'hôtes donnée.
#
# Variables requises :
#   - vault_root_path : Chemin racine dans Vault
#   - target_host : Liste des hôtes cibles (séparés par des virgules)
#   - target_user : Liste des utilisateurs cibles (séparés par des virgules)
#   - timestamp : Horodatage pour les noms de fichiers
#   - zip_password : Mot de passe pour le fichier ZIP
#
# Structure du traitement :
#   1. Initialisation des variables
#   2. Récupération des mots de passe
#   3. Affichage des résultats
#   4. Export sécurisé
#   5. Création du ZIP chiffré
#   6. Déplacement vers les artéfacts
#   7. Nettoyage
#   8. Stockage du mot de passe ZIP
#   9. Journalisation
###############################################################################

###############################################################################
# 1. Initialisation
###############################################################################

# Initialisation des variables de suivi pour le processus de récupération
- name: Initialize variables
  set_fact:
    vault_root_path: "{{ vault_root_path }}"
    success_combinations: []  # Liste des combinaisons hôte/utilisateur réussies
    failed_combinations: []   # Liste des combinaisons hôte/utilisateur en échec
    host_user_passwords: {}   # Stockage temporaire des mots de passe par hôte/utilisateur
    hosts_list: "{{ target_host.split(',') }}"    # Conversion en liste des hôtes
    users_list: "{{ target_user.split(',') }}"    # Conversion en liste des utilisateurs

###############################################################################
# 2. Récupération des mots de passe
###############################################################################

# Boucle sur chaque combinaison possible hôte/utilisateur
# Utilise le produit cartésien pour tester toutes les combinaisons
- name: Retrieve password for each host/user combination
  include_tasks: retrieve_password_for_host.yml
  loop: "{{ hosts_list | product(users_list) | list }}"  # Génère toutes les combinaisons
  loop_control:
    loop_var: current_combination  # Format: [hostname, username]
```

ANNEXES

Annexe 3

main_vault_retrieve.yml - Orchestration récupération

```
################################################################################
# 3. Affichage des résultats
################################################################################

# Affichage des résultats par combinaison
- name: Confirm successful retrievals
  debug:
    msg: "Mot de passe récupéré avec succès pour {{ item.1 }} sur {{ item.0 }}"
  loop: "{{ success_combinations }}"

# Liste des échecs
- name: List failed combinations
  debug:
    msg: "Échec de récupération du mot de passe pour {{ item.1 }} sur {{ item.0 }}"
  when: failed_combinations | length > 0
  loop: "{{ failed_combinations }}"

# Résumé global
- name: Summary of password retrieval
  debug:
    msg: >-
      Résumé:
      Total des combinaisons: {{ hosts_list | length * users_list | length }},
      Récupérations réussies: {{ success_combinations | length }},
      Échecs: {{ failed_combinations | length }}

################################################################################
# 4. Export sécurisé des mots de passe
################################################################################

# Création du fichier CSV avec formatage strict
- name: Create password file
  copy:
    content: |
      hostname,username,password
      {% for combo in success_combinations %}
      {{ combo[0] }},{{ combo[1] }},{{ host_user_passwords[combo[0] ~ '_' ~ combo[1]] }}
      {% endfor %}
    dest: "/tmp/passwords_{{ timestamp }}.csv"
    mode: '0600'
  when: success_combinations | length > 0
  register: password_file
```

ANNEXES

Annexe 3

main_vault_retrieve.yml - Orchestration récupération

```
#####################################################################
# 5. Création et sécurisation du ZIP
#####################################################################

# Création du ZIP chiffré seulement si des mots de passe ont été récupérés
- name: Create encrypted ZIP
  block:
    # Vérification de l'existence du fichier source
    - name: Check if password file exists
      stat:
        path: "{{ password_file.dest | default('') }}"
      register: pwd_file_stat
      when: password_file is defined and password_file.changed

    # Création du ZIP avec mot de passe
    - name: Create ZIP file
      shell: |
        cd /tmp && zip -P "{{ zip_password }}" "passwords_{{ timestamp }}.zip" "passwords_{{ timestamp }}.csv"
      args:
        executable: /bin/sh
      when: pwd_file_stat.stat is defined and pwd_file_stat.stat.exists
      register: zip_creation

    # Suppression immédiate du fichier temporaire
    - name: Remove password file
      file:
        path: "{{ password_file.dest }}"
        state: absent
      when: password_file is defined and password_file.dest is defined and zip_creation is defined and zip_creation is success
  when: success_combinations | length > 0

# Affichage d'un avertissement si aucun mot de passe n'a été récupéré
- name: Show warning if no passwords were retrieved
  debug:
    msg: |
      Information : Aucun mot de passe n'a été récupéré. Hôtes concernés : {{ failed_combinations | join(', ') }} | Note : Ceci n'est pas une erreur, il est possible que les secrets n'existent pas pour ces hôtes.
  when: success_combinations | length == 0

#####################################################################
# 6. Déplacement vers les artéfacts
#####################################################################

# Déplacement du ZIP vers le répertoire des artéfacts
- name: Move ZIP to artifacts directory
  block:
    # Vérification de l'existence du ZIP
    - name: Check if ZIP exists
      stat:
        path: "/tmp/passwords_{{ timestamp }}.zip"
      register: zip_file_stat

    # Création du répertoire des artéfacts
    - name: Create artifacts directory
      file:
        path: "{{ lookup('env', 'CI_PROJECT_DIR') }}"
        state: directory
        mode: '0755'
      when: zip_file_stat.stat.exists

    # Copie du ZIP vers les artéfacts
    - name: Copy ZIP to artifacts
      copy:
        src: "/tmp/passwords_{{ timestamp }}.zip"
        dest: "{{ lookup('env', 'CI_PROJECT_DIR') }}/passwords.zip"
        remote_src: yes
        mode: '0600'  # Permissions restrictives sur le ZIP final
      when: zip_file_stat.stat.exists
      register: zip_moved

#####################################################################
# 7. Nettoyage
#####################################################################

# Suppression des fichiers temporaires
- name: Clean up temporary files
  file:
    path: "/tmp/passwords_{{ timestamp }}.zip"
    state: absent
  when: zip_moved is defined and zip_moved.changed

#####################################################################
# 8. Envoi du mot de passe ZIP dans le Vault
#####################################################################

- name: Setup ZIP password in Vault
  include_tasks: setup_zip_password.yml
  when: success_combinations | length > 0

#####################################################################
# 9. Journalisation
#####################################################################

- name: Log operation
  include_tasks: log_operation.yml
  vars:
    action: "retrieve"
    operation_status: "{{ 'SUCCESS (' ~ success_combinations | length ~ ' passwords)' if success_combinations | length > 0 else 'NO_PASSWORDS_FOUND' }}"
```

ANNEXES

Annexe 3

retrieve_password_for_host.yml - Récupération par hôte

```yaml
# Objectif : Ce module est appelé pour chaque hôte et tente de récupérer
# le mot de passe correspondant dans Vault. Il met à jour les variables de suivi
# pour garder trace des succès et des échecs.
#
# Paramètres attendus :
# - current_combination[0] : Nom de l'hôte cible (remplace current_host)
# - current_combination[1] : Nom de l'utilisateur (remplace current_target_user)
# - tenant_id : Identifiant du tenant dans Vault
# - vault_root_path : Chemin racine dans Vault
#
# Variables mises à jour :
# - host_user_passwords : Dictionnaire stockant les mots de passe récupérés
# - success_combinations : Liste des combinaisons réussies
# - failed_combinations : Liste des combinaisons en échec
###############################################################################

# Bloc principal pour la récupération du mot de passe avec gestion d'erreurs
- name: Try to retrieve password from Vault
  block:
    # Affiche le chemin complet du secret dans Vault pour le débogage
    # Format : vault_root_path/CAV/tenant_id/VMs/hostname/users_system/username
    - name: Check Vault path
      debug:
        msg: "Tentative d'accès au secret: {{ vault_root_path  }}/CAV/{{ tenant_id }}/VMs/{{ current_combination[0] }}/users_system/{{ current_combination[1] }}"

    # Utilisation du module community.hashi_vault pour lire le secret
    # auth_method: token -> utilise le token Vault pour l'authentification
    # no_log: true -> masque les informations sensibles des logs
    # failed_when: false -> continue même si le secret n'existe pas
    - name: Get password from Vault
      community.hashi_vault.vault_read:
        path: "{{ vault_root_path }}/CAV/{{ tenant_id }}/VMs/{{ current_combination[0] }}/users_system/{{ current_combination[1] }}"
        auth_method: token
      register: vault_result
      no_log: true
      failed_when: false

    # Vérifie que la réponse de Vault contient bien un secret valide
    # Stocke le résultat dans vault_success pour utilisation ultérieure
    - name: Verify Vault result
      set_fact:
        vault_success: "{{ vault_result is succeeded and vault_result.data is defined and vault_result.data.data is defined }}"
      no_log: true

    # Si le secret existe, extrait le mot de passe du résultat
    # Le mot de passe est stocké dans data.data.password
    - name: Extract password from Vault result
      set_fact:
        current_password: "{{ vault_result.data.data.password }}"
      no_log: true
      when: vault_success | bool

    # Si l'extraction a réussi :
    # 1. Stocke le mot de passe dans le dictionnaire host_user_passwords
    # 2. Ajoute la combinaison à la liste des succès
    - name: Store password and mark success
      set_fact:
        host_user_passwords: "{{ host_user_passwords | combine({{current_combination[0] ~ '_' ~ current_combination[1]: current_password}) }}"
        success_combinations: "{{ success_combinations + [current_combination] }}"
      when: vault_success | bool

  # Bloc rescue : s'exécute si une erreur survient dans le bloc principal
  rescue:
    # Ajoute la combinaison à la liste des échecs pour le rapport final
    - name: Mark combination as failed
      set_fact:
        failed_combinations: "{{ failed_combinations + [current_combination] }}"
```

ANNEXES

Annexe 3

setup_zip_password.yml - Gestion ZIP et sécurité

```yaml
###############################################################################
# Module de gestion du mot de passe ZIP dans HashiCorp Vault
#
# Description:
#   Ce module stocke de manière sécurisée le mot de passe utilisé pour chiffrer
#   le fichier ZIP contenant les mots de passe récupérés. Le mot de passe est
#   stocké dans un chemin dédié dans Vault avec un horodatage unique.
#
# Variables requises:
#   - vault_root_path: Chemin racine dans Vault
#   - tenant_id: Identifiant du tenant
#   - timestamp: Horodatage pour identifier uniquement le mot de passe
#   - zip_password: Mot de passe généré pour le ZIP
#
# Sécurité:
#   - Utilise l'authentification par token Vault
#   - Stocke le mot de passe dans un chemin sécurisé et unique
#   - Vérifie le succès de l'écriture dans Vault
###############################################################################

# Stockage sécurisé du mot de passe ZIP dans Vault
# Le chemin inclut un horodatage pour garantir l'unicité
- name: Stocker le mot de passe dans Vault
  community.hashi_vault.vault_write:
    path: "{{ vault_root_path }}/CAV/{{ tenant_id }}/Vault_Manager/Vault_Password_Retrieval_System/password_{{ timestamp }}"
    data:
      password: "{{ zip_password }}"    # Stocke le mot de passe généré
    auth_method: token                  # Utilise l'authentification par token
  register: vault_write                 # Enregistre le résultat pour vérification

# Vérification du succès de l'opération
# Affiche un message de confirmation avec le chemin de stockage
- name: Afficher la confirmation
  debug:
    msg: "Mot de passe ZIP stocké avec succès dans {{ vault_root_path }}/CAV/{{ tenant_id }}/Vault_Password_Retrieval_System/password_{{ timestamp }}"
  when: vault_write is succeeded
```

ANNEXES

Annexe 3

main_vault_delete.yml - Orchestration suppression

```yaml
################################################################################
# Module de suppression des secrets dans HashiCorp Vault
#
# Description:
#   Ce module gère la suppression des secrets dans Vault pour toutes les
#   combinaisons hôte/utilisateur spécifiées. Il maintient des compteurs
#   pour suivre le statut de chaque opération.
#
# Variables requises:
#   - target_host: Liste des hôtes cibles (séparés par des virgules)
#   - target_user: Liste des utilisateurs cibles (séparés par des virgules)
#
# Métriques suivies:
#   - success_count: Nombre de suppressions réussies
#   - not_found_count: Nombre de secrets non trouvés
#   - failed_count: Nombre d'échecs de suppression
################################################################################

# Étape 1: Préparation des listes d'hôtes et d'utilisateurs
# Conversion des chaînes en listes pour le traitement
- name: Préparation des listes
  set_fact:
    hosts_list: "{{ target_host.split(',') }}"    # Conversion de la chaîne d'hôtes en liste
    users_list: "{{ target_user.split(',') }}"    # Conversion de la chaîne d'utilisateurs en liste

# Étape 2: Initialisation des compteurs de suivi
# Ces compteurs permettront de générer un rapport final détaillé
- name: Initialisation des compteurs
  set_fact:
    success_count: 0     # Nombre de secrets supprimés avec succès
    failed_count: 0      # Nombre d'échecs de suppression
    not_found_count: 0   # Nombre de secrets non trouvés

# Étape 3: Traitement de chaque combinaison hôte/utilisateur
# Utilisation du produit cartésien pour générer toutes les combinaisons possibles
- name: Traitement de chaque combinaison
  include_tasks: delete_password_for_host.yml  # Appel du fichier de suppression individuelle
  loop: "{{ hosts_list | product(users_list) | list }}"  # Génère toutes les combinaisons
  loop_control:
    loop_var: current_combination  # Format: [hostname, username]

# Étape 4: Affichage du rapport final
# Présentation d'un résumé détaillé des opérations effectuées
- name: Afficher le résumé des opérations
  debug:
    msg: >-
      Résumé des opérations :
      Total des combinaisons traitées : {{ (hosts_list | length) * (users_list | length) }} |
      Suppressions réussies : {{ success_count }} |
      Secrets non trouvés : {{ not_found_count }} |
      Échecs de suppression : {{ failed_count }}

# Étape 5: Journalisation
- name: Log operation
  include_tasks: log_operation.yml
  vars:
    action: "delete"
    operation_status: "SUCCESS ({{ success_count }} deleted, {{ not_found_count }} not found)"
```

ANNEXES

Annexe 3

delete_password_for_host.yml - suppression par hôte

```yaml
################################################################################
# Module de suppression des secrets dans HashiCorp Vault
#
# Description:
#   Ce module gère la suppression des secrets dans Vault pour toutes les
#   combinaisons hôte/utilisateur spécifiées. Il maintient des compteurs
#   pour suivre le statut de chaque opération.
#
# Variables requises:
#   - target_host: Liste des hôtes cibles (séparés par des virgules)
#   - target_user: Liste des utilisateurs cibles (séparés par des virgules)
#
# Métriques suivies:
#   - success_count: Nombre de suppressions réussies
#   - not_found_count: Nombre de secrets non trouvés
#   - failed_count: Nombre d'échecs de suppression
################################################################################

# Étape 1: Préparation des listes d'hôtes et d'utilisateurs
# Conversion des chaînes en listes pour le traitement
- name: Préparation des listes
  set_fact:
    hosts_list: "{{ target_host.split(',') }}"    # Conversion de la chaîne d'hôtes en liste
    users_list: "{{ target_user.split(',') }}"    # Conversion de la chaîne d'utilisateurs en liste

# Étape 2: Initialisation des compteurs de suivi
# Ces compteurs permettront de générer un rapport final détaillé
- name: Initialisation des compteurs
  set_fact:
    success_count: 0     # Nombre de secrets supprimés avec succès
    failed_count: 0      # Nombre d'échecs de suppression
    not_found_count: 0   # Nombre de secrets non trouvés

# Étape 3: Traitement de chaque combinaison hôte/utilisateur
# Utilisation du produit cartésien pour générer toutes les combinaisons possibles
- name: Traitement de chaque combinaison
  include_tasks: delete_password_for_host.yml  # Appel du fichier de suppression individuelle
  loop: "{{ hosts_list | product(users_list) | list }}"  # Génère toutes les combinaisons
  loop_control:
    loop_var: current_combination  # Format: [hostname, username]

# Étape 4: Affichage du rapport final
# Présentation d'un résumé détaillé des opérations effectuées
- name: Afficher le résumé des opérations
  debug:
    msg: >-
      Résumé des opérations :
      Total des combinaisons traitées : {{ (hosts_list | length) * (users_list | length) }} |
      Suppressions réussies : {{ success_count }} |
      Secrets non trouvés : {{ not_found_count }} |
      Échecs de suppression : {{ failed_count }}

# Étape 5: Journalisation
- name: Log operation
  include_tasks: log_operation.yml
  vars:
    action: "delete"
    operation_status: "SUCCESS ({{ success_count }} deleted, {{ not_found_count }} not found)"
```

ANNEXES

Annexe 3

main_vault_check_delete.yml - Orchestration check suppression

```yaml
#########################################################################
# Module de vérification avant suppression des secrets dans HashiCorp Vault
#
# Description:
#   Ce module effectue une vérification préalable des secrets avant leur
#   suppression dans Vault. Il génère un rapport détaillé sur les secrets
#   trouvés et non trouvés pour chaque combinaison hôte/utilisateur.
#
# Variables requises:
#   - vault_root_path: Chemin racine dans Vault
#   - target_host: Liste des hôtes cibles (séparés par des virgules)
#   - target_user: Liste des utilisateurs cibles (séparés par des virgules)
#
# Fonctionnement:
#   1. Initialisation des variables de suivi
#   2. Vérification de chaque combinaison hôte/utilisateur
#   3. Génération d'un rapport détaillé
#   4. Affichage des avertissements nécessaires
#########################################################################

#########################################################################
# 1. Initialisation des variables de suivi
# - Préparation des listes pour le suivi des vérifications
# - Conversion des chaînes d'entrée en listes exploitables
#########################################################################

- name: Initialize check variables
  set_fact:
    vault_root_path: "{{ vault_root_path }}"
    exists_hosts: []     # Liste des hôtes où au moins un utilisateur existe
    not_found_hosts: []  # Liste des hôtes sans aucun utilisateur trouvé
    hosts_list: "{{ target_host.split(',') }}"    # Conversion en liste des hôtes
    users_list: "{{ target_user.split(',') }}"    # Conversion en liste des utilisateurs

#########################################################################
# 2. Vérification de chaque combinaison
# - Utilisation du produit cartésien pour tester toutes les combinaisons
# - Appel du fichier de vérification individuelle pour chaque paire
#########################################################################

- name: Check users existence on each host
  include_tasks: check_users_for_host.yml
  loop: "{{ hosts_list | product(users_list) | list }}"  # Génère toutes les combinaisons possibles
  loop_control:
    loop_var: current_combination  # Format: [hostname, username]

#########################################################################
# 3. Génération du rapport détaillé
# - Calcul des hôtes sans secrets
# - Affichage du résumé global
# - Détail par machine
#########################################################################

# Identification des hôtes sans secrets trouvés
- name: Calculate not found hosts
  set_fact:
    not_found_hosts: "{{ hosts_list | difference(exists_hosts) }}"

# Titre du rapport
- name: Afficher le titre
  debug:
    msg: "RÉSUMÉ DE LA VÉRIFICATION DE SUPPRESSION"

# Liste des utilisateurs concernés
- name: Afficher les utilisateurs cibles
  debug:
    msg: "Utilisateurs à supprimer : {{ users_list | join(', ') }}"

# Rapport détaillé par machine avec utilisateurs trouvés
- name: Afficher le résumé par machine
  debug:
    msg: "MACHINE : {{ host }} | UTILISATEURS PRÉSENTS : {% for combo in host_user_combinations %}{% if combo[0] == host %}{{ combo[1] }}{% if not loop.last %}, {% endif %}{% endif %}{% endfor %}"
  loop: "{{ exists_hosts | unique }}"
  loop_control:
    loop_var: host
  when: exists_hosts | length > 0
  vars:
    host_user_combinations: "{{ lookup('vars', 'host_user_combinations', default=[]) }}"

# Message d'avertissement et instructions pour la suite
- name: Afficher avertissement
  debug:
    msg: "Pour procéder à la suppression, utilisez l'étape suivante du pipeline. | La suppression concernera uniquement les machines où les utilisateurs existent."
```

ANNEXES

Annexe 3

check_users_for_host.yml - check par hôte

```yaml
################################################################################
# Module de vérification des secrets utilisateurs dans HashiCorp Vault
#
# Description:
#   Ce fichier vérifie l'existence des secrets dans Vault pour une combinaison
#   donnée d'hôte et d'utilisateur. Il met à jour les listes de suivi pour
#   les secrets trouvés.
#
# Variables requises:
#   - current_combination[0]: Nom de l'hôte à vérifier
#   - current_combination[1]: Nom de l'utilisateur à vérifier
#   - vault_root_path: Chemin racine dans Vault
#   - tenant_id: Identifiant du tenant
################################################################################

# Affichage d'un message de debug pour suivre la progression de la vérification
# Cette tâche permet de visualiser quelle combinaison est en cours de traitement
- name: Display checking message
  debug:
    msg: "Vérification de l'utilisateur {{ current_combination[1] }} sur {{ current_combination[0] }}"

# Vérification de l'existence du secret dans Vault
# Utilise le module community.hashi_vault pour interroger l'API Vault
# Le paramètre no_log assure que les données sensibles ne sont pas journalisées
# failed_when: false permet de continuer même si le secret n'existe pas
- name: Verify secret in Vault
  community.hashi_vault.vault_read:
    path: "{{ vault_root_path }}/CAV/{{ tenant_id }}/VMs/{{ current_combination[0] }}/users_system/{{ current_combination[1] }}"
    auth_method: token
  register: vault_check
  no_log: true
  failed_when: false

# Mise à jour des listes de suivi si le secret existe
# exists_hosts: Liste des hôtes où au moins un secret existe
# host_user_combinations: Liste des combinaisons hôte/utilisateur valides
# Cette mise à jour n'est effectuée que si le secret est trouvé dans Vault
- name: Update host and combination lists
  set_fact:
    exists_hosts: "{{ exists_hosts + [current_combination[0]] }}"
    host_user_combinations: "{{ host_user_combinations | default([]) + [current_combination] }}"
  when: vault_check is succeeded and vault_check.data is defined
```

ANNEXES

Annexe 3

log_operation.yml - systeme de log

```
################################################################################
# Module de journalisation des opérations
################################################################################

- name: Create logs directory
  file:
    path: "{{ lookup('env', 'CI_PROJECT_DIR') }}/logs"
    state: directory
    mode: '0755'

- name: Generate log entry content
  set_fact:
    log_entry: |
      {{ timestamp }} | {{ action | upper }} | Pipeline #{{ lookup('env', 'CI_PIPELINE_ID') }}
      {% if action == 'retrieve' %}
      Hôtes avec accès trouvés:
      {% for host in success_combinations | default([]) %}
      - {{ host[0] }} (utilisateur: {{ host[1] }})
      {% endfor %}
      {% else %}
      {% if deleted_hosts is defined and deleted_hosts | length > 0 %}
      Secrets supprimés avec succès:
      {% for host in deleted_hosts %}
      - {{ host[0] }} (utilisateur: {{ host[1] }})
      {% endfor %}
      {% endif %}
      {% if not_found_count | int > 0 %}
      Machines sans secret : {{ not_found_count }}
      {% endif %}
      Tenant: {{ tenant_id }}
      Status: {{ operation_status }} (détails: {{ success_count }} supprimé(s))
      {% endif %}
      ----------------------------------------

- name: Ensure log file exists
  file:
    path: "{{ lookup('env', 'CI_PROJECT_DIR') }}/logs/operations.log"
    state: touch
    mode: '0644'

- name: Add new log entry
  blockinfile:
    path: "{{ lookup('env', 'CI_PROJECT_DIR') }}/logs/operations.log"
    block: "{{ log_entry }}"
    insertbefore: BOF
    create: yes
    marker: "# {mark} PIPELINE {{ lookup('env', 'CI_PIPELINE_ID') }}"
    mode: '0644'
```

ANNEXES

Annexe 4 - gitlab-ci.yml - fichier pour le pipeline CI-CD de gitlab

```
################################################################################
# Pipeline GitLab CI/CD pour la récupération sécurisée des mots de passe
#
# Objectif : Ce pipeline permet de récupérer de manière sécurisée les mots de
# passe stockés dans HashiCorp Vault pour une liste d'hôtes donnée.
#
# Sécurité :
# - Authentification forte avec Vault
# - Gestion sécurisée des clés SSH
# - Protection des données sensibles
################################################################################

# Template de déploiement - Installation des outils et dépendances
.deploy: &deploy
  # Installation des paquets système nécessaires
  - apk add sshpass vault libcap zip bash openssl python3 py3-pip tzdata
  - setcap cap_ipc_lock= /usr/sbin/vault  # Autorise Vault à verrouiller la mé moire

  # Configuration Python et dépendances
  - pip3 install --upgrade pip
  - pip3 install hvac  # Client Python pour Vault

  # Installation des modules Ansible requis
  - ansible-galaxy collection install community.hashi_vault

  # Configuration Git pour accès sécurisé aux dépôts
  - apk update && apk add git
  - git config --global url."https://gitlab-ci-token:$CI_JOB_TOKEN@sourcehub.orange-business.com".insteadOf https://sourcehub.orange-business.com

# Template d'authentification - Configuration des accès sécurisés
.authentification: &authentification
  # Configuration Vault avec authentification JWT
  - export VAULT_ADDR="https://vault.factory.orange-business.com:8200"
  - export VAULT_TOKEN="$(vault write -field=token auth/sourcehub/login role=project-${CI_PROJECT_ID} jwt=${VAULT_ID_TOKEN})"

  # Configuration SSH avec clé privée stockée dans Vault
  - export SSH_PRIVATE_KEY="$(vault read -field=MYRUNNER_SSH_PRIVATE_KEY ${VAULT_ROOT_PATH}/CAV/${CAV_ID_TENANT}/VMs/obs_myrunner)"
  - 'which ssh-agent || ( apk update && apk add openssh-client )'
  - eval $(ssh-agent -s)
  - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add - > /dev/null

  # Sécurisation du répertoire SSH
  - mkdir -p ~/.ssh
  - chmod 700 ~/.ssh

  # Configuration de l'accès à l'inventaire
  - export CUSTOMER_INVENTORY_PROJECT=${CI_SERVER_URL}/${CI_PROJECT_ROOT_NAMESPACE}/inventory.git

# Configuration de l'image de base
.image_and_tags:
  image: cytopia/ansible:latest  # Image avec Ansible préinstallé
```

ANNEXES

Annexe 4 - gitlab-ci.yml - fichier pour le pipeline CI-CD de gitlab

```
# Variables globales du pipeline
variables:
  ACTION:
    value: "retrieve"  # Action par défaut
    description: "Action à exécuter (retrieve, delete)"
  CI_TARGET_HOST:
    value: ""  # Vide par défaut, à remplir lors de l'exécution
    description: "Nom de la machine cible"
  CI_TARGET_USER:
    value: "osadmin"  # Utilisateur par défaut
    description: "Nom de l'utilisateur cible"
  CI_TARGET_GROUP:
    value: "all"  # Groupe par défaut
    description: "Groupes d'inventaire à cibler (séparés par des virgules: windows,linux,DB)"
  ANSIBLE_ROOT: ${CI_PROJECT_DIR}/ansible  # Répertoire racine Ansible
  VAULT_ROOT_PATH: 'prj_${CI_PROJECT_ROOT_NAMESPACE}'  # Chemin racine dans Vault

# Scripts d'initialisation exécutés avant chaque job
before_script:
    - *deploy  # Installation des outils
    - *authentification  # Configuration des accès

    # Configuration Ansible
    - export ANSIBLE_FORCE_COLOR=true
    - export ANSIBLE_HOST_KEY_CHECKING=False
    - export TZ=Europe/Paris
    - export TIMESTAMP=$(date '+%d-%m-%Y_%Hh%M')
    # Récupération de l'inventaire dynamique
    - git clone ${CUSTOMER_INVENTORY_PROJECT} environment
    - cd environment

    # Configuration des cibles
    - INVENTORY_FILE="${CI_PROJECT_DIR}/environment/00hosts"
    - if [ ! -f "$INVENTORY_FILE" ]; then echo "Le fichier 00hosts n'existe pas dans le dépôt d'inventaire"; exit 1; fi
    - export TARGET_USER="${CI_TARGET_USER:-osadmin}"  # Utilisateur par défaut: osadmin

    # Extraction de la liste des hôtes depuis l'inventaire avec gestion multi-groupes
    - export TARGET_GROUP="${CI_TARGET_GROUP:-all}"
    - echo "Groupes ciblés = ${TARGET_GROUP}"
    - >
      INVENTORY_FILE="00hosts";
      group_list=$(echo "$TARGET_GROUP" | tr ',' ' ');
      hosts_list="";
      for g in $group_list; do
        group_hosts=$(awk -v grp="$g" '
          $0 == "["grp"]" {flag=1; next}
          /^\[/ {flag=0}
          flag && NF>=1 {print $1}
        ' "$INVENTORY_FILE");
        if [ -n "$group_hosts" ]; then
          hosts_list="$hosts_list $group_hosts";
        fi;
      done;
      hosts_list=$(echo "$hosts_list" | tr ' ' '\n' | sort -u | grep -v '^$');
      if [ -z "$hosts_list" ]; then
        echo "[ERREUR] Aucun hôte trouvé pour les groupes: $TARGET_GROUP";
        exit 1;
      fi;
      TARGET_HOSTS=$(echo "$hosts_list" | tr '\n' ',' | sed 's/,$//')

    - echo "RÉSUMÉ DE LA RECHERCHE"
    - 'echo "- Groupes demandés : $TARGET_GROUP"'
    - 'echo "- Liste des hôtes  : $TARGET_HOSTS"'
    - 'export TARGET_HOST="${CI_TARGET_HOST:-$TARGET_HOSTS}"'
    - echo "Interrogation des machines = ${TARGET_HOST}"
    - echo "Utilisateur = ${TARGET_USER}"
```

ANNEXES

Annexe 4 - gitlab-ci.yml - fichier pour le pipeline CI-CD de gitlab

```yaml
# Définition des étapes du pipeline
stages:
  - retrieve_password
  - check_delete
  - confirm_delete

# Configuration globale du cache
cache:
  key: "pipeline-logs-${CI_PROJECT_ID}"  # Clé unique par projet
  paths:
    - logs/
  when: always
  policy: pull-push

# Job de récupération des mots de passe
retrieve_vm_passwords:
  stage: retrieve_password
  id_tokens:
    VAULT_ID_TOKEN:
      aud: $CI_SERVER_URL
  extends: .image_and_tags

  cache:
    key: "pipeline-logs-${CI_PROJECT_ID}"
    paths:
      - logs/
    policy: pull-push
    when: always

  artifacts:
    paths:
      - passwords.zip
      - logs/operations.log
    expire_in: 1 day  # Suppression après 1 jour

  # Exécution du rôle Ansible pour la récupération
  script:
    - cd ${ANSIBLE_ROOT}
    - export ZIP_PASSWORD=$(openssl rand -base64 32)
    - >-
      ansible-playbook vault_password_manager.yml
      -e "action=retrieve"
      -e "timestamp=${TIMESTAMP}"
      -e "zip_password=${ZIP_PASSWORD}"
      -e "tenant_id=${CAV_ID_TENANT}"
      -e "vault_root_path=${VAULT_ROOT_PATH}"
      -e "target_host=${TARGET_HOST}"
      -e "target_user=${TARGET_USER}"

  rules:
    - if: '$ACTION == "retrieve" || $ACTION == ""'
      when: manual
```

```yaml
# Job de vérification avant suppression
check_vm_password:
  stage: check_delete
  id_tokens:
    VAULT_ID_TOKEN:
      aud: $CI_SERVER_URL
  extends: .image_and_tags
  script:
    - cd ${ANSIBLE_ROOT}
    - >-
      ansible-playbook vault_password_manager.yml
      -e "action=check_delete"
      -e "tenant_id=${CAV_ID_TENANT}"
      -e "vault_root_path=${VAULT_ROOT_PATH}"
      -e "target_host=${TARGET_HOST}"
      -e "target_user=${TARGET_USER}"
  rules:
    - if: '$ACTION == "delete"'

# Job de confirmation et suppression
confirm_vm_password:
  stage: confirm_delete
  id_tokens:
    VAULT_ID_TOKEN:
      aud: $CI_SERVER_URL
  extends: .image_and_tags
  cache:
    key: "pipeline-logs-${CI_PROJECT_ID}"
    paths:
      - logs/
    policy: pull-push
    when: always

  artifacts:
    paths:
      - logs/operations.log
    expire_in: 1 day  # Suppression après 1 jour
  script:
    - cd ${ANSIBLE_ROOT}
    - >-
      ansible-playbook vault_password_manager.yml
      -e "action=delete"
      -e "timestamp=${TIMESTAMP}"
      -e "tenant_id=${CAV_ID_TENANT}"
      -e "vault_root_path=${VAULT_ROOT_PATH}"
      -e "target_host=${TARGET_HOST}"
      -e "target_user=${TARGET_USER}"
  rules:
    - if: '$ACTION == "delete"'
      when: manual
```

ANNEXES

Annexe 5 - gitlab-ci.yml - fichier pour le pipeline CI-CD de gitlab

```yaml
# Définition des étapes du pipeline
stages:
  - retrieve_password
  - check_delete
  - confirm_delete

# Configuration globale du cache
cache:
  key: "pipeline-logs-${CI_PROJECT_ID}"  # Clé unique par projet
  paths:
    - logs/
  when: always
  policy: pull-push

# Job de récupération des mots de passe
retrieve_vm_passwords:
  stage: retrieve_password
  id_tokens:
    VAULT_ID_TOKEN:
      aud: $CI_SERVER_URL
  extends: .image_and_tags

  cache:
    key: "pipeline-logs-${CI_PROJECT_ID}"
    paths:
      - logs/
    policy: pull-push
    when: always

  artifacts:
    paths:
      - passwords.zip
      - logs/operations.log
    expire_in: 1 day  # Suppression après 1 jour

  # Exécution du rôle Ansible pour la récupération
  script:
    - cd ${ANSIBLE_ROOT}
    - export ZIP_PASSWORD=$(openssl rand -base64 32)
    - >-
      ansible-playbook vault_password_manager.yml
      -e "action=retrieve"
      -e "timestamp=${TIMESTAMP}"
      -e "zip_password=${ZIP_PASSWORD}"
      -e "tenant_id=${CAV_ID_TENANT}"
      -e "vault_root_path=${VAULT_ROOT_PATH}"
      -e "target_host=${TARGET_HOST}"
      -e "target_user=${TARGET_USER}"

  rules:
    - if: '$ACTION == "retrieve" || $ACTION == ""'
      when: manual
```

```yaml
# Job de vérification avant suppression
check_vm_password:
  stage: check_delete
  id_tokens:
    VAULT_ID_TOKEN:
      aud: $CI_SERVER_URL
  extends: .image_and_tags
  script:
    - cd ${ANSIBLE_ROOT}
    - >-
      ansible-playbook vault_password_manager.yml
      -e "action=check_delete"
      -e "tenant_id=${CAV_ID_TENANT}"
      -e "vault_root_path=${VAULT_ROOT_PATH}"
      -e "target_host=${TARGET_HOST}"
      -e "target_user=${TARGET_USER}"
  rules:
    - if: '$ACTION == "delete"'

# Job de confirmation et suppression
confirm_vm_password:
  stage: confirm_delete
  id_tokens:
    VAULT_ID_TOKEN:
      aud: $CI_SERVER_URL
  extends: .image_and_tags
  cache:
    key: "pipeline-logs-${CI_PROJECT_ID}"
    paths:
      - logs/
    policy: pull-push
    when: always

  artifacts:
    paths:
      - logs/operations.log
    expire_in: 1 day  # Suppression après 1 jour
  script:
    - cd ${ANSIBLE_ROOT}
    - >-
      ansible-playbook vault_password_manager.yml
      -e "action=delete"
      -e "timestamp=${TIMESTAMP}"
      -e "tenant_id=${CAV_ID_TENANT}"
      -e "vault_root_path=${VAULT_ROOT_PATH}"
      -e "target_host=${TARGET_HOST}"
      -e "target_user=${TARGET_USER}"
  rules:
    - if: '$ACTION == "delete"'
      when: manual
```

ANNEXES

Annexe 5 - Documentation technique complète

Documentation Détaillée - Système de Gestion des Mots de Passe Vault

Vue d'ensemble

Système sécurisé de récupération et suppression des mots de passe de VM stockés dans HashiCorp Vault, utilisant des pipelines GitLab CI/CD et des rôles Ansible modulaires.

Architecture Technique

1. Composants Principaux

- HashiCorp Vault :
  - Stockage sécurisé des secrets
  - Gestion fine des accès par JWT
  - Version supportée : ≥ 1.12.0
  - Authentification par token et JWT
- GitLab CI/CD :
  - Orchestration des pipelines
  - Gestion des artifacts
  - Stages de validation
  - Variables d'environnement sécurisées
- Ansible :
  - Automatisation et rôles modulaires
  - Collection community.hashi_vault
  - Gestion idempotente des opérations
  - Templates Jinja2 pour les rapports
- OpenSSL :
  - Génération cryptographique des mots de passe ZIP
  - Algorithmes : AES-256-CBC
  - Longueur clés : 256 bits
  - Format de sortie : Base64

2. Structure des Rôles

```
vault_password_manager/
├── README.md
├── defaults/
│   └── main.yml                    # Variables par défaut du rôle
├── handlers/
│   └── main.yml                    # Handlers du rôle
├── meta/
│   └── main.yml                    # Métadonnées Ansible Galaxy
├── tasks/
│   ├── main.yml                    # Point d'entrée, gestion des actions
│   ├── main_vault_retrieve.yml     # Orchestration récupération
│   ├── main_vault_delete.yml       # Orchestration suppression
│   ├── main_vault_check_delete.yml # Vérification pré-suppression
│   ├── retrieve_password_for_host.yml  # Récupération par hôte
│   ├── delete_password_for_host.yml    # Suppression par hôte
│   ├── check_users_for_host.yml        # Vérification par hôte
│   ├── setup_zip_password.yml          # Gestion ZIP et mots de passe
│   └── log_operation.yml               # Journalisation des opérations
├── vars/
│   └── main.yml                    # Variables internes du rôle
└── tests/
    ├── inventory                   # Inventaire de test
    └── test.yml                    # Playbook de test
```

ANNEXES

Annexe 5 - Documentation technique complète

Fonctionnalités Détaillées

1. Récupération des Mots de Passe

- Processus automatisé en plusieurs étapes :
  1. Authentification Vault (JWT)
  2. Vérification des droits d'accès
  3. Génération mot de passe ZIP (OpenSSL)
  4. Récupération des secrets
  5. Export CSV sécurisé
  6. Création ZIP chiffré
  7. Stockage mot de passe ZIP dans Vault
  8. Nettoyage des fichiers temporaires

2. Suppression des Secrets

- Processus sécurisé en deux stages :
  - Stage 1 : Vérification
    1. Scan des secrets existants
    2. Génération rapport détaillé
    3. Liste des impacts potentiels
  - Stage 2 : Suppression
    1. Confirmation manuelle requise
    2. Suppression sécurisée
    3. Rapport final détaillé

Configuration Détaillée

1. Variables GitLab CI/CD

Obligatoires :

Variable	Description	Exemple
`CAV_ID_TENANT`	ID du tenant	`cav01av01ocb0000023`

Optionnelles :

Variable	Description	Défaut
`CI_TARGET_USER`	Utilisateurs	`osadmin`
`CI_TARGET_HOST`	Hôtes cibles	`all`
`CI_TARGET_GROUP`	Groupes	`all`
`ACTION`	Opération	`retrieve`

2. Structure Vault

```
project/
├── ansible/
│   ├── ansible.cfg
│   ├── vault_password_manager.yml
│   ├── roles/
│   │   └── vault_password_manager/
│   │       ├── README.md
│   │       ├── defaults/
│   │       │   └── main.yml
│   │       ├── handlers/
│   │       │   └── main.yml
│   │       ├── meta/
│   │       │   └── main.yml
│   │       ├── tasks/
│   │       │   ├── main.yml
│   │       │   ├── main_vault_retrieve.yml
│   │       │   ├── main_vault_delete.yml
│   │       │   ├── main_vault_check_delete.yml
│   │       │   ├── retrieve_password_for_host.yml
│   │       │   ├── delete_password_for_host.yml
│   │       │   ├── check_users_for_host.yml
│   │       │   ├── setup_zip_password.yml
│   │       │   └── log_operation.yml
│   │       ├── vars/
│   │       │   └── main.yml
│   │       └── tests/
│   │           ├── inventory
│   │           └── test.yml
├── .gitlab-ci.yml
└── docs/
```

ANNEXES

Annexe 5 - Documentation technique complète

Guide d'Utilisation Détaillé

1. Récupération des Mots de Passe

```
# Via interface GitLab CI/CD
1. Accéder au pipeline
2. Sélectionner ACTION=retrieve
3. Spécifier les cibles (hôtes/groupes/utilisateurs)
4. Lancer le pipeline
5. Récupérer l'artifact ZIP dans les résultats du pipeline
6. Récupérer le mot de passe ZIP dans Vault
```

2. Suppression des Secrets (Processus en 2 étapes)

```
# Étape 1 : Vérification
1. Accéder au pipeline
2. Sélectionner ACTION=delete
3. Spécifier les cibles (hôtes/groupes/utilisateurs)
4. Lancer le pipeline
5. Examiner le rapport de vérification automatique qui liste :
   - Les secrets trouvés par hôte
   - Les utilisateurs concernés
   - Le nombre total de secrets à supprimer

# Étape 2 : Confirmation
6. Le pipeline s'arrête automatiquement à l'étape de confirmation
7. Examiner une dernière fois le rapport
8. Cliquer sur le bouton "Play" de étape de suppression pour confirmer
9. Consulter le rapport final de suppression :
   - Nombre de suppressions réussies
   - Nombre de secrets non trouvés
   - Liste des éventuelles erreurs
```

Sécurité et Conformité

1. Protection des Données

- Authentification JWT avec Vault
- Chiffrement AES-256 des ZIP
- Permissions restrictives (0600)
- Nettoyage automatique
- Audit complet des accès

2. Bonnes Pratiques

- Principe du moindre privilège
- Isolation des tâches
- Validation multi-niveaux
- Traçabilité complète
- Documentation exhaustive

Rapports et Logs

1. Format des Exports

```
hostname,username,password
host1,user1,xxxxx
host2,user2,xxxxx
```

2. Rapports de Suppression

- Nombre de succès/échecs
- Liste détaillée par hôte
- Horodatage des opérations
- Utilisateurs impactés

Maintenance et Support

1. Dépendances

- HashiCorp Vault
- GitLab Enterprise/Community
- Ansible Core ≥ 2.12
- Python ≥ 3.8
- OpenSSL ≥ 1.1.1

2. Résolution des Problèmes

- Logs détaillés dans GitLab
- Traçage des opérations Vault
- Messages d'erreur explicites
- Documentation des cas courants

Processus Techniques Détaillés

1. Génération des Mots de Passe ZIP

```
# Commande OpenSSL utilisée
openssl rand -base64 32 | cut -c1-20
```

2. Format du Fichier CSV

```
hostname,username,password
host1.domain.com,user1,xxxxx
# Séparateur: virgule (,)
# Encodage: UTF-8
# BOM: Non
```

3. Structure des Chemins Vault

```
${VAULT_ROOT_PATH}/
├── CAV/
│   └── ${CAV_ID_TENANT}/
│       ├── VMs/
│       │   └── ${VM_NAME}/
│       │       └── users_system/
│       │           └── ${USERNAME}
│       └── Vault_Password_Retrieval_System/
│           └── password_${TIMESTAMP}
```

4. Mécanismes de Sécurité

- Permissions fichiers :
  - CSV temporaire : 0600
  - ZIP : 0644
  - Clés SSH : 0400
- Nettoyage :
  - Suppression immédiate post-export
  - Force avec `shred -u`
  - Vérification d'existence

ANNEXES

Annexe 6 - Justification Technique du Projet

Choix Techniques

1. Utilisation d'Ansible avec Rôles

Pourquoi Ansible ?

- **Modularité et Maintenabilité:**
  - Organisation en rôles distincts pour une meilleure lisibilité
  - Séparation des responsabilités par tâches spécifiques
  - Facilité de maintenance et d'évolution du code
  - Structure standardisée et documentée
- **Intégration Native:**
  - Module community.hashi_vault disponible
  - Gestion efficace des tâches séquentielles

2. Sécurisation des Exports

Format ZIP Chiffré

- **Justification:**
  - Protection double couche :
    - ZIP protégé par mot de passe
    - Mot de passe stocké dans Vault
  - Facilité d'utilisation pour l'utilisateur final
  - Format standard compatible avec tous les systèmes
- **Avantages:**
  - Suppression automatique des fichiers temporaires
  - Isolation des données sensibles
  - Traçabilité des accès via Vault
  - Solution légère et efficace

3. Pipeline de Suppression en Deux Stages

Pourquoi Cette Architecture ?

- **Sécurité Opérationnelle:**
  - Prévention des suppressions accidentelles
  - Validation humaine obligatoire
  - Rapport détaillé avant confirmation
  - Possibilité d'annulation
- **Avantages du Processus:**

  1. Stage de Vérification :
     - Liste exhaustive des secrets à supprimer
     - Évaluation de l'impact
     - Pas de modification effective
  2. Stage de Suppression :
     - Confirmation explicite requise
     - Opération irréversible clairement identifiée
     - Rapport final détaillé

ANNEXES

Annexe 6 - Justification Technique du Projet

Impact des Choix Techniques

Bénéfices

1. **Maintenance:**
   - Code modulaire et réutilisable
   - Documentation intégrée
   - Structure claire et évolutive
2. **Sécurité:**
   - Protection des données exportées
   - Processus de suppression sécurisé
   - Traçabilité des opérations
3. **Expérience Utilisateur:**
   - Interface GitLab CI/CD familière
   - Processus clair et documenté
   - Rapports détaillés à chaque étape

Limitations Acceptées

- Dépendance à l'infrastructure Vault existante
- Nécessité d'accès GitLab
- Stockage temporaire des mots de passe en ZIP

ANNEXES

Annexe 7 - Glossaire détaillé

Ansible : Outil d'automatisation open-source utilisé pour la gestion de la configuration, le déploiement d'applications, la gestion des tâches, et l'orchestration des services IT.

HashiCorp Vault : Outil de gestion des secrets et de protection des données sensibles, offrant un stockage sécurisé et un accès contrôlé aux informations sensibles.

GitLab CI/CD : Plateforme de développement et d'intégration continue/déploiement continu (CI/CD) qui permet de gérer le cycle de vie complet du développement logiciel, de la planification à la livraison.

DevOps : Ensemble de pratiques visant à améliorer la collaboration entre les équipes de développement (Dev) et d'exploitation (Ops) pour automatiser et accélérer le processus de livraison des logiciels.

Cloud Computing : Modèle de prestation de services informatiques via Internet, permettant un accès à la demande à des ressources informatiques partagées.

Cybersécurité : Ensemble des technologies, processus et pratiques conçus pour protéger les réseaux, les appareils, les programmes et les données contre les attaques, les dommages ou les accès non autorisés.

AES-256 : Standard de chiffrement avancé utilisant des clés de 256 bits pour sécuriser les données.

Traçabilité : Capacité à suivre et documenter l'historique, l'utilisation ou la localisation d'un article ou d'une activité.

DevSecOps : Intégration des pratiques de sécurité dans le processus DevOps pour garantir que la sécurité est prise en compte tout au long du cycle de vie du développement logiciel.

Gestion des secrets : Processus de gestion des informations sensibles comme les mots de passe, les clés API, etc., de manière sécurisée.

Pipeline CI/CD : Série d'étapes automatisées qui permettent de construire, tester et déployer du code de manière continue et cohérente.

Infrastructure virtuelle : Environnement informatique simulé ou émulé, souvent utilisé pour exécuter des machines virtuelles et des services.

Chiffrement : Processus de transformation des données en un format illisible sans une clé de déchiffrement spécifique.

Automatisation : Utilisation de la technologie pour effectuer des tâches sans intervention humaine.

SSH (**Secure Shell**) : Protocole de réseau cryptographique pour l'exploitation sécurisée des services réseau sur un réseau non sécurisé.

WSL (**Windows Subsystem for Linux**) : Fonctionnalité de Windows qui permet d'exécuter un environnement Linux directement sur Windows.

Visual Studio Code : Éditeur de code source développé par Microsoft pour Windows, Linux et macOS. Il inclut le support pour le débogage, l'intégration Git, la coloration syntaxique, etc.

Logs d'audit : Enregistrements détaillés des activités et des événements pour permettre le suivi et l'analyse.

Gestion des erreurs : Processus de détection, de journalisation et de gestion des erreurs pour assurer la robustesse des applications.

Modularité : Conception d'un système en modules indépendants qui peuvent être développés, testés et maintenus séparément.

ANNEXES

Annexe 7 - Glossaire détaillé

Intégration continue (CI) : Pratique de développement logiciel où les développeurs intègrent régulièrement leur code dans un dépôt partagé, suivi de builds et de tests automatisés.

Déploiement continu (CD) : Extension de l'intégration continue où le code est automatiquement déployé en production après avoir passé les tests.

OpenSSL : Outil de cryptographie open-source qui fournit des fonctions de cryptographie, de chiffrement et de gestion des certificats SSL/TLS.

Rôle Ansible : Ensemble de tâches, de gestionnaires, de fichiers, de modèles et de variables qui peuvent être réutilisés pour automatiser des tâches spécifiques.

Playbook Ansible : Fichier YAML contenant une série de tâches à exécuter par Ansible pour configurer des systèmes ou déployer des applications.

Automatisation des tâches : Utilisation de scripts, d'outils et de logiciels pour effectuer des tâches répétitives sans intervention humaine.

Gestion des artefacts : Processus de gestion des fichiers générés lors du développement et du déploiement des logiciels, tels que les binaires, les bibliothèques, etc.

Traçabilité des opérations : Capacité à suivre et à enregistrer toutes les opérations effectuées sur un système pour assurer la transparence et la responsabilité.

I**ntégration avec HashiCorp Vault** : Processus de connexion et d'utilisation des services de gestion des secrets de HashiCorp Vault dans un environnement automatisé.

Gestion des accès et permissions : Processus de contrôle et de gestion des droits d'accès aux ressources et aux systèmes pour assurer la sécurité et la conformité.

ANNEXES

Annexe 7 - Glossaire détaillé

Intégration continue (CI) : Pratique de développement logiciel où les développeurs intègrent régulièrement leur code dans un dépôt partagé, suivi de builds et de tests automatisés.

Déploiement continu (CD) : Extension de l'intégration continue où le code est automatiquement déployé en production après avoir passé les tests.

OpenSSL : Outil de cryptographie open-source qui fournit des fonctions de cryptographie, de chiffrement et de gestion des certificats SSL/TLS.

Rôle Ansible : Ensemble de tâches, de gestionnaires, de fichiers, de modèles et de variables qui peuvent être réutilisés pour automatiser des tâches spécifiques.

Playbook Ansible : Fichier YAML contenant une série de tâches à exécuter par Ansible pour configurer des systèmes ou déployer des applications.

Automatisation des tâches : Utilisation de scripts, d'outils et de logiciels pour effectuer des tâches répétitives sans intervention humaine.

Gestion des artefacts : Processus de gestion des fichiers générés lors du développement et du déploiement des logiciels, tels que les binaires, les bibliothèques, etc.

Traçabilité des opérations : Capacité à suivre et à enregistrer toutes les opérations effectuées sur un système pour assurer la transparence et la responsabilité.

I**ntégration avec HashiCorp Vault** : Processus de connexion et d'utilisation des services de gestion des secrets de HashiCorp Vault dans un environnement automatisé.

Gestion des accès et permissions : Processus de contrôle et de gestion des droits d'accès aux ressources et aux systèmes pour assurer la sécurité et la conformité.

Annexe 8 - Bibliographie avec liens

Ouvrages et documentation technique

- HashiCorp. (2024). Vault Documentation.
  - https://developer.hashicorp.com/vault/docs
  - Documentation officielle du gestionnaire de secrets HashiCorp Vault, utilisée comme base pour la conception de la solution sécurisée de gestion et transmission des mots de passe d'infrastructure.

- Ansible Documentation Team. (2024). Ansible User Guide.
  - https://docs.ansible.com/ansible/latest/user_guide/index.html
  - Documentation officielle d'Ansible utilisée pour le développement des rôles d'automatisation, y compris le module community.hashi_vault.

- GitLab Inc. (2024). GitLab CI/CD Pipelines Documentation.
  - https://docs.gitlab.com/ee/ci/
  - Guide officiel pour l'architecture et la sécurisation des pipelines GitLab CI/CD, utilisé pour l'intégration de l'automatisation continue avec Ansible et Vault.

Standards et bonnes pratiques

- International Organization for Standardization. (2013). ISO/IEC 27001:2013 – Information security management systems.
  - Résumé des exigences normatives relatives à la sécurité de l'information, dont la gestion des accès et la traçabilité, appliquées dans la conception du système.

- International Organization for Standardization. (2013). ISO/IEC 27002:2013 – Code of practice for information security controls.
  - Liste des bonnes pratiques pour le contrôle des accès aux informations sensibles et la gestion sécurisée des identifiants.

Annexe 8 - Bibliographie avec liens

Articles, guides et ressources en ligne

- OpenSSL Software Foundation. (2023). OpenSSL: Cryptography and SSL/TLS Toolkit Documentation.
  - https://www.openssl.org/docs/
  - Documentation OpenSSL, utilisée pour la gestion du chiffrement des archives ZIP contenant les mots de passe.

- Community.hashi_vault Ansible Collection Documentation. (2023).
  - https://docs.ansible.com/ansible/latest/collections/community/hashi_vault/
  - Guide pour l'intégration d'Ansible et Vault via les modules communautaires officiels.

Ressources internes et retours d'expérience

- Orange Business, Global Delivery & Operations Team. (2024). Entretiens internes sur la gestion collaborative des mots de passe (interviews avec l'équipe destinataire).
  - Interviews menées pour recueillir les besoins métiers, les attentes sur l'automatisation et la sécurité de la transmission des secrets.

Outils et logiciels

- HashiCorp Vault – Version 1.12+
  - Outil open source déployé pour la gestion centralisée et sécurisée des secrets.

- Ansible Core – Version 2.12+
  - Plateforme d'automatisation utilisée pour la création des playbooks et rôles techniques.

- GitLab Enterprise – CI/CD Pipelines
  - Plateforme DevOps utilisée pour automatiser l'exécution des tâches d'infrastructure et la sécurisation de la transmission des secrets.

Outils d'assistance par intelligence artificielle

- Perplexity AI. (2024). Moteur de recherche augmenté par l'intelligence artificielle
  - Utilisé pour la recherche documentaire, la confrontation des sources et la veille technologique.
- Anthropic. (2024). Claude – Modèle conversationnel d'aide à la rédaction et à la programmation
  - Employé pour la clarification, la reformulation technique et l'assistance rédactionnelle.