



Automatisation avec Ansible pour la Sécurité et la Surveillance

Une approche systématique pour renforcer et surveiller un'infrastructure

Job 01 - Prise en Main et Premières Automatisations

1

Installation

Installer Ansible sur une machine de contrôle et configurer l'environnement nécessaire

2

Configuration SSH

Mettre en place une authentification SSH sans mot de passe entre le contrôleur et les machines cibles

3

Inventaire

Créer un fichier d'inventaire structuré pour organiser les machines gérées

4

Automatisation

Développer et exécuter des playbooks pour automatiser les tâches de base



Installation et configuration

- Installation d'Ansible via `apt install ansible` sur Debian
- Génération de clés SSH avec `ssh-keygen` et déploiement avec `ssh-copy-id`
- Création d'un inventaire avec deux machines (192.168.40.156 et 192.168.40.158) dans le groupe [workers]

```
GNU nano 7.2                                         inventory.ini *
[workers]
192.168.40.156
192.168.40.158

[all_servers:children]
workers|
```

Playbooks développés

- Vérification du service SSH
- Tester la communication avec les machines.

```
dome@Ansible-Master:~$ ansible all -i inventory.ini -m ping
192.168.40.156 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.40.158 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
dome@Ansible-Master:~$ |
```

- Copie de fichiers entre le contrôleur et les cibles `copy_file.yml`
- Mise à jour des paquets adaptée au système d'exploitation `update_packages.yml`

```
dome@Ansible-Master:~$ ansible-playbook -i inventory.ini check_service.yml
PLAY [all] ****
TASK [Gathering Facts] ****
ok: [192.168.40.156]
ok: [192.168.40.158]

TASK [Vérifier si le service SSH est actif] ****
ok: [192.168.40.158]
ok: [192.168.40.156]

PLAY RECAP ****
192.168.40.156 : ok=2    changed=0    unreachable=0    failed=0    skipped=0
                  rescued=0   ignored=0
192.168.40.158 : ok=2    changed=0    unreachable=0    failed=0    skipped=0
                  rescued=0   ignored=0
dome@Ansible-Master:~$ |
```

Difficultés rencontrées

Échec des commandes avec sudo

Le problème était lié aux priviléges requis pour certaines opérations.

Solution (pas sécurisé): Création d'un fichier dans /etc/sudoers.d/ansible_user avec l'option NOPASSWD pour permettre l'élévation de priviléges sans mot de passe.

Job 02 - Hardening des Systèmes

- Suppression des utilisateurs système inutiles (games, irc, etc.)

```
dome@Ansible-Master:~$ ansible-playbook -i inventory.ini clear_users.yml
PLAY [Supprimer les utilisateurs système inutiles] ****
TASK [Gathering Facts] ****
ok: [192.168.40.158]
ok: [192.168.40.156]

TASK [Supprimer chaque utilisateur non essentiel] ****
changed: [192.168.40.156] => (item=games)
changed: [192.168.40.158] => (item=games)
changed: [192.168.40.158] => (item=irc)
changed: [192.168.40.156] => (item=irc)
changed: [192.168.40.158] => (item=news)
changed: [192.168.40.156] => (item=news)
changed: [192.168.40.158] => (item=uucp)
changed: [192.168.40.156] => (item=uucp)
changed: [192.168.40.158] => (item=lp)
changed: [192.168.40.156] => (item=lp)
changed: [192.168.40.158] => (item=proxy)
changed: [192.168.40.156] => (item=proxy)
changed: [192.168.40.158] => (item=list)
changed: [192.168.40.156] => (item=list)
ok: [192.168.40.158] => (item=gnats)
ok: [192.168.40.156] => (item=gnats)

PLAY RECAP ****
192.168.40.156 : ok=2    changed=1    unreachable=0    failed=0
192.168.40.158 : ok=2    changed=1    unreachable=0    failed=0

dome@Ansible-Master:~$ |
```

- Création du compte sécurisé ansible_admin avec mot de passe haché (SHA-512)

```
- name: Création de l'utilisateur administrateur ansible_admin
  hosts: all
  become: yes
  vars:
    ansible_admin_password: "$6$flwYeggW/27bCpUL$8ex9W1kZv6Wt6vNUQhZVyBamftuHhaoUNXnm1h."
  tasks:
    - name: Créer l'utilisateur ansible_admin
      user:
        name: ansible_admin
        comment: "Utilisateur dédié à Ansible"
        shell: /bin/bash
        password: "{{ ansible_admin_password }}"
        groups: "{{ 'sudo' if ansible_facts['os_family'] == 'Debian' else 'wheel' }}"
        append: yes
```

- Durcissement SSH : désactivation de root, authentification par clé uniquement, port 2002
- Pare-feu UFW : blocage par défaut, autorisation explicite des ports 2002, 80, 443
- Désactivation de services non essentiels (cups, avahi-daemon)



Job 03 - Surveillance, Logs et Conformité

Mettre en place une **surveillance active** des serveurs Linux, auditer leur conformité, et renforcer la sécurité via l'automatisation Ansible.

Déploiement de Filebeat

- **But :** Collecter automatiquement les logs système et sécurité
- **Méthode :** Installation automatisée via Ansible

```
dome@Ansible-Master:~/playbook$ ansible-playbook -i inventory.ini filebeat.yml

PLAY [Installer et configurer Filebeat] ****
TASK [Gathering Facts] ****
ok: [192.168.40.158]
ok: [192.168.40.156]

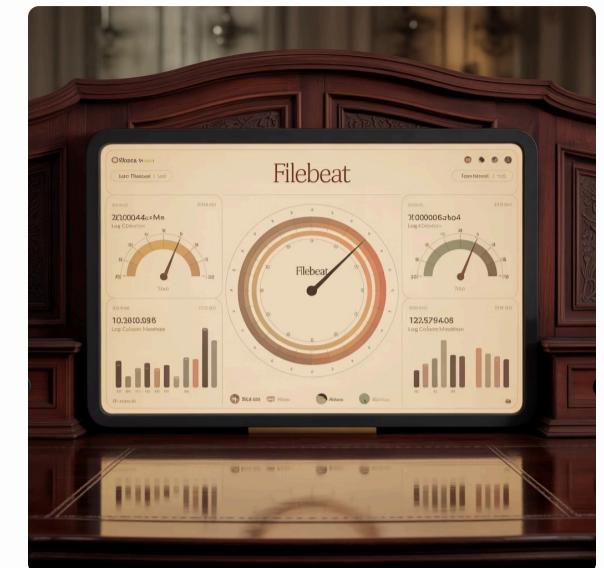
TASK [Installer gpg si nécessaire] ****
ok: [192.168.40.158]
ok: [192.168.40.156]

TASK [Ajouter la clé GPG d'Elastic] ****
ok: [192.168.40.158]
ok: [192.168.40.156]

TASK [Ajouter le dépôt Filebeat] ****
ok: [192.168.40.156]
ok: [192.168.40.158]

TASK [Mettre à jour les paquets] ****
ok: [192.168.40.156]
ok: [192.168.40.158]

TASK [Installer Filebeat] ****
ok: [192.168.40.156]
ok: [192.168.40.158]
```



Exemple de tâche Ansible :

```
- name: Installer Filebeat
apt:
  name: filebeat
  state: present
```



Pour tester l'infrastructure sans impacter un environnement de production, nous avons configuré une simulation d'envoi des logs vers un serveur fictif.

Configuration de sortie dans le fichier filebeat.yml :

Cette simulation permet de vérifier que la collecte fonctionne correctement avant d'intégrer avec l'infrastructure de surveillance réelle.

```
output.logstash:
  hosts: ["192.0.2.1:5044"]
```

- *Comme il n'est pas demandé de déployer un serveur de destination réel (ex: Logstash ou Elasticsearch), la consigne est de simuler la collecte des logs.*

Cela signifie que l'on configure Filebeat comme s'il envoyait les logs, mais vers une adresse fictive, ou vers une destination locale.

L'IP 192.0.2.1 fait partie des adresses réservées à la documentation (RFC 5737), donc idéale pour une **simulation sans risque**.

Alternativement, pour une sortie locale (utile pour vérifier que Filebeat fonctionne), on peut aussi utiliser la sortie **vers un fichier local** :

```
output.file:
  path: "/tmp/filebeat-output"
  filename: "filebeat.log"
```

- *Avant d'ajouter la clé GPG, il faut s'assurer que l'outil GnuPG (gpg) est installé sur chaque machine.*

```
- name: Installer et configurer Filebeat
hosts: all
become: yes
tasks:
  - name: Installer gpg si nécessaire
    package:
      name: gnupg
      state: present

  - name: Ajouter la clé GPG d'Elastic
    apt_key:
      url: https://artifacts.elastic.co/GPG-KEY-elasticsearch
      state: present
```

Template Jinja2 utilisé

Un template Jinja2 (filebeat.yml.j2) a été utilisé pour générer dynamiquement le fichier de configuration de Filebeat. Ce fichier doit être placé dans le dossier templates/ du projet Ansible.

Avantages du template Jinja2

- Personnalisation dynamique selon l'environnement
- Configuration centralisée et facilement modifiable
- Réutilisation des patterns de configuration
- Adaptabilité aux différentes versions de systèmes

Exemple de template filebeat.yml.j2 :

```
filebeat.inputs:  
- type: log  
paths:  
- /var/log/*.log  
  
output.logstash:  
hosts: ["{{ logstash_server }}:{{  
logstash_port }}"]  
  
{% if enable_system_module %}  
filebeat.modules:  
- module: system  
enabled: true  
{% endif %}
```

Résultat de la mise en place de Filebeat

Collecte centralisée

Tous les logs système sont désormais collectés automatiquement sans intervention manuelle sur chaque serveur.

Configuration uniforme

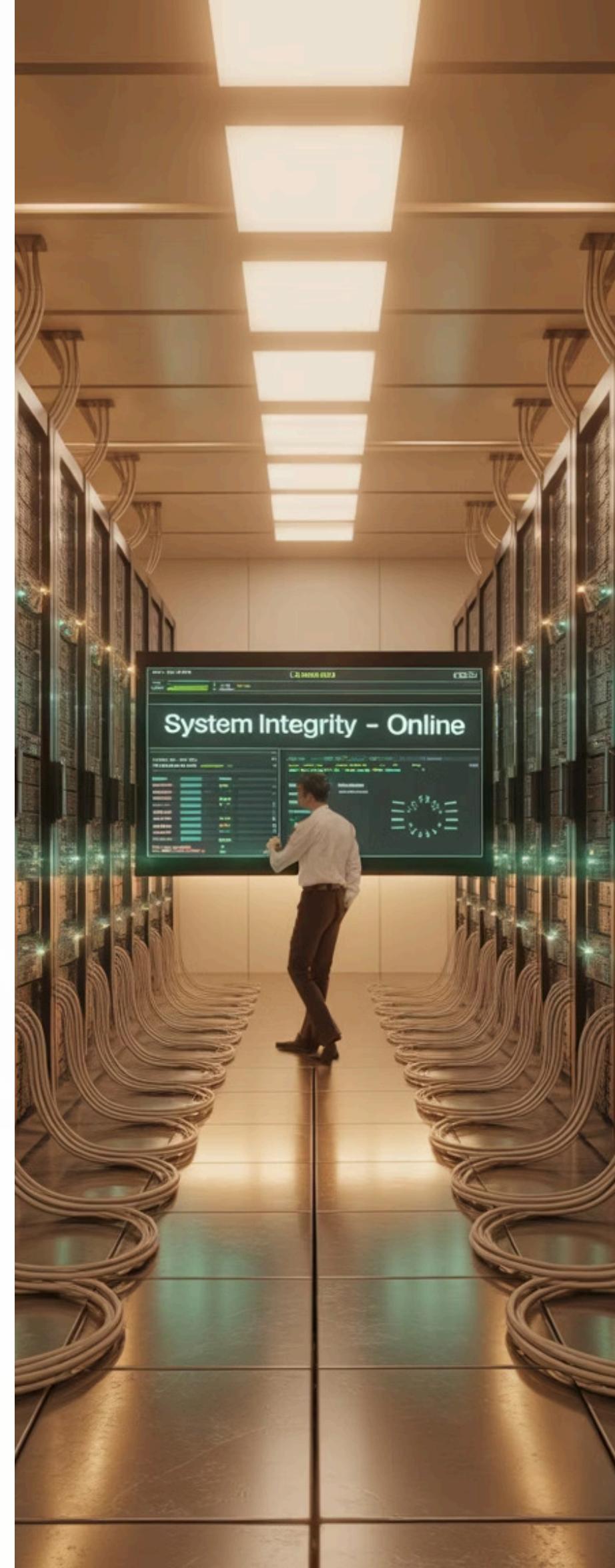
Application cohérente des paramètres de collecte sur l'ensemble du parc de serveurs.

Adaptabilité

Structure modulaire permettant d'activer facilement de nouvelles sources de logs selon les besoins.

Maintenance simplifiée

Les mises à jour de configuration peuvent être déployées rapidement via Ansible sur tous les serveurs.



Activation des modules system et audited

L'activation des modules permet de recueillir des informations cruciales sur les événements système et les audits de sécurité. Ces données sont essentielles pour :

- L'existence et les permissions des fichiers critiques. Vérifier les permissions de /etc/passwd, /etc/shadow, /etc/sudoers, /etc/ssh/sshd_config.
- Lister les utilisateurs ayant des mots de passe vides.
- Rechercher des répertoires 777.

```
dome@Ansible-Master:~/playbook$ ansible-playbook -i inventory.ini audit_conformite.yml
PLAY [Audit de conformité de base] *****
TASK [Gathering Facts] *****
ok: [192.168.40.156]
ok: [192.168.40.158]

TASK [Vérifier les permissions de /etc/passwd] *****
ok: [192.168.40.158]
ok: [192.168.40.156]

TASK [Afficher les permissions de /etc/passwd] *****
ok: [192.168.40.156] => {
    "msg": "Permissions /etc/passwd : 0644"
}
ok: [192.168.40.158] => {
    "msg": "Permissions /etc/passwd : 0644"
}

TASK [Vérifier les permissions de /etc/shadow] *****
ok: [192.168.40.156]
ok: [192.168.40.158]

TASK [Afficher les permissions de /etc/shadow] *****
ok: [192.168.40.156] => {
    "msg": "Permissions /etc/shadow : 0640"
}
ok: [192.168.40.158] => {
    "msg": "Permissions /etc/shadow : 0640"
}

TASK [Vérifier les permissions de /etc/sudoers] *****
ok: [192.168.40.156]
ok: [192.168.40.158]
```

```
TASK [Afficher les permissions de /etc/sudoers] *****
ok: [192.168.40.156] => {
    "msg": "Permissions /etc/sudoers : 0440"
}
ok: [192.168.40.158] => {
    "msg": "Permissions /etc/sudoers : 0440"
}

TASK [Vérifier les permissions de /etc/ssh/sshd_config] *****
ok: [192.168.40.156]
ok: [192.168.40.158]

TASK [Afficher les permissions de /etc/ssh/sshd_config] *****
ok: [192.168.40.156] => {
    "msg": "Permissions /etc/ssh/sshd_config : 0644"
}
ok: [192.168.40.158] => {
    "msg": "Permissions /etc/ssh/sshd_config : 0644"
}

TASK [Rechercher les utilisateurs avec mot de passe vide] *****
ok: [192.168.40.158]
ok: [192.168.40.156]

TASK [Afficher les utilisateurs sans mot de passe] *****
ok: [192.168.40.156] => {
    "utilisateurs_sans_mdp.stdout_lines": []
}
ok: [192.168.40.158] => {
    "utilisateurs_sans_mdp.stdout_lines": []
}

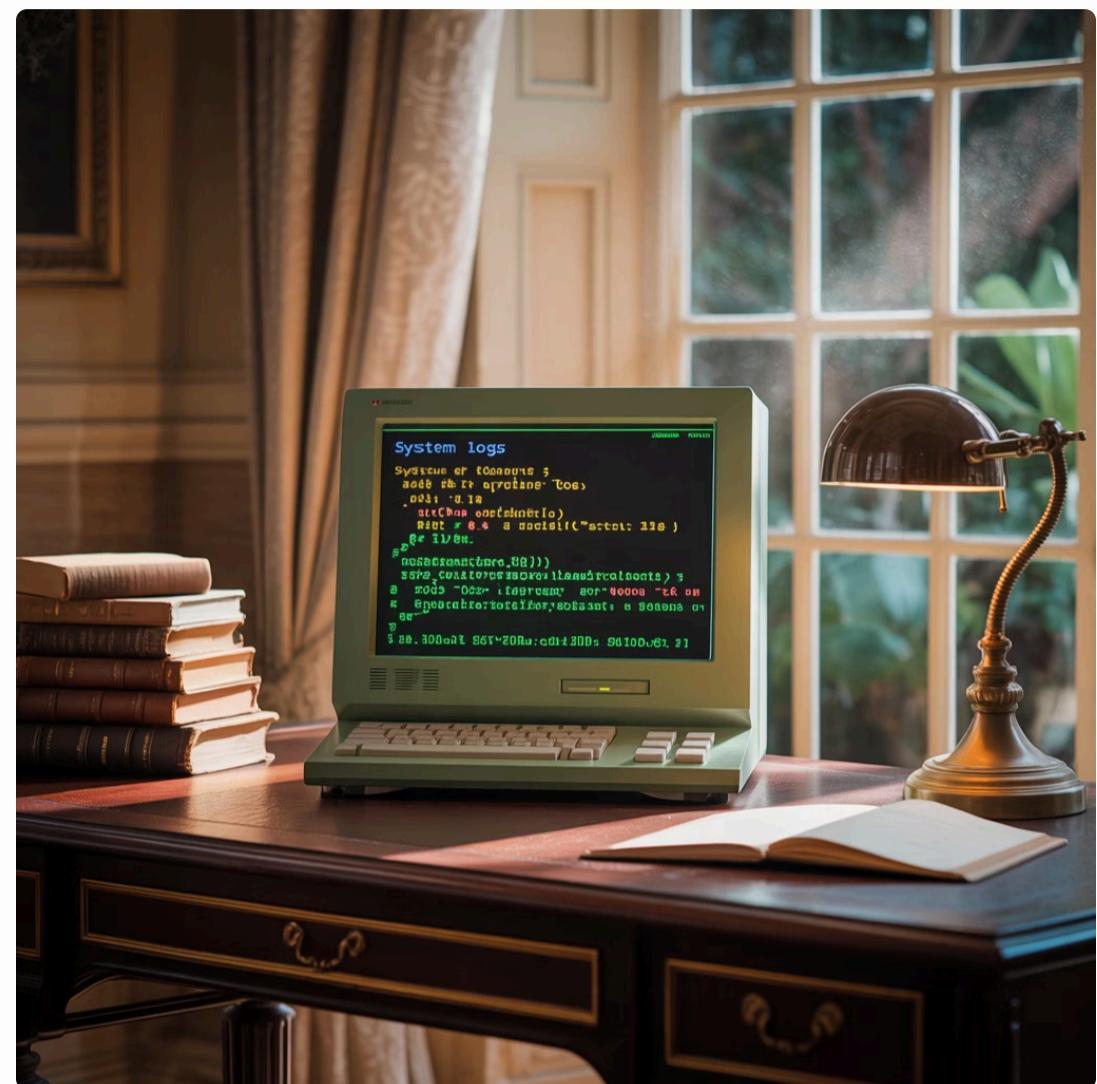
TASK [Rechercher les répertoires avec permission 777] *****
ok: [192.168.40.158]
ok: [192.168.40.156]
```

```
TASK [Rechercher les répertoires avec permission 777] *****
ok: [192.168.40.158]
ok: [192.168.40.156]

TASK [Afficher les répertoires 777] *****
ok: [192.168.40.156] => {
    "repertoires_777.stdout_lines": []
}
ok: [192.168.40.158] => {
    "repertoires_777.stdout_lines": []
}

PLAY RECAP *****
192.168.40.156      : ok=13    changed=0    unreachable=0
192.168.40.158      : ok=13    changed=0    unreachable=0

dome@Ansible-Master:~/playbook$ |
```



Exemple Ansible

```
- name: Vérifier les permissions de /etc/passwd
stat:
path: /etc/passwd
register: passwd_stat

- name: Alerter si les permissions sont incorrectes
debug:
msg: "ALERTE: /etc/passwd a des permissions incorrectes"
when: passwd_stat.stat.mode != '0644'

- name: Trouver les utilisateurs sans mot de passe
shell: awk -F: '($2 == "") {print $1}' /etc/shadow
register: empty_passwords
changed_when: false
```

Contrôles effectués

- Vérification des permissions critiques
 - /etc/passwd
 - /etc/shadow
 - /etc/sudoers
- Détection des utilisateurs sans mot de passe
- Recherche de répertoires trop permisifs (chmod 777)

Durcissement du Noyau et des Politiques de Mot de Passe

```
dome@Ansible-Master:~/playbook$ ansible-playbook -i inventory.ini durcissement.yml
PLAY [Durcissement du noyau et des politiques de mot de passe] ****
TASK [Gathering Facts] ****
ok: [192.168.40.158]
ok: [192.168.40.156]

TASK [Activer les cookies TCP SYN pour limiter les attaques DoS] ****
changed: [192.168.40.158]
changed: [192.168.40.156]

TASK [Désactiver le routage IP] ****
changed: [192.168.40.156]
changed: [192.168.40.158]

TASK [Forcer la longueur minimale des mots de passe dans /etc/login.defs] ****
changed: [192.168.40.156]
changed: [192.168.40.158]

TASK [Forcer la durée de validité des mots de passe] ****
changed: [192.168.40.156]
changed: [192.168.40.158]

TASK [Interdire la réutilisation des anciens mots de passe (PAM)] ****
changed: [192.168.40.156]
changed: [192.168.40.158]

PLAY RECAP ****
192.168.40.156      : ok=6    changed=5    unreachable=0    failed=0    skipped=0
192.168.40.158      : ok=6    changed=5    unreachable=0    failed=0    skipped=0

dome@Ansible-Master:~/playbook$ |
```

Détails importants

- `sysctl` applique immédiatement les paramètres réseau de sécurité.
- `lineinfile` assure que les valeurs dans `/etc/login.defs` soient forcées (longueur, validité).
- **PAM** (via `common-password`) empêche la réutilisation des 5 derniers mots de passe.

⚠ Si la distribution n'utilise pas `/etc/pam.d/common-password` (cas de certaines RedHat/CentOS), il faut adapter vers `/etc/pam.d/system-auth`.

Paramètres du noyau

```
- name: Désactiver le routage IP
  sysctl:
    name: net.ipv4.ip_forward
    value: "0"
    state: present

- name: Protéger contre les attaques ICMP
  sysctl:
    name: net.ipv4.icmp_echo_ignore_all
    value: "1"
    state: present
```

Objectif

- Appliquer des paramètres de sécurité réseau dans `/etc/sysctl.conf`.
- Modifier les politiques de mot de passe via PAM ou `login.defs`.

Étapes détaillées

- Ajouter des paramètres de type `net.ipv4.tcp_syncookies = 1`.
- Appliquer les modifications avec la commande `sysctl -p`.

Politique de mots de passe

```
- name: Longueur minimale des mots de passe
  lineinfile:
    path: /etc/login.defs
    regexp: '^PASS_MIN_LEN'
    line: 'PASS_MIN_LEN 12'

- name: Durée de vie maximale
  lineinfile:
    path: /etc/login.defs
    regexp: '^PASS_MAX_DAYS'
    line: 'PASS_MAX_DAYS 90'
```

Introduction aux Rôles Ansible :

Le renforcement de la sécurité par l'automatisation

Objectif

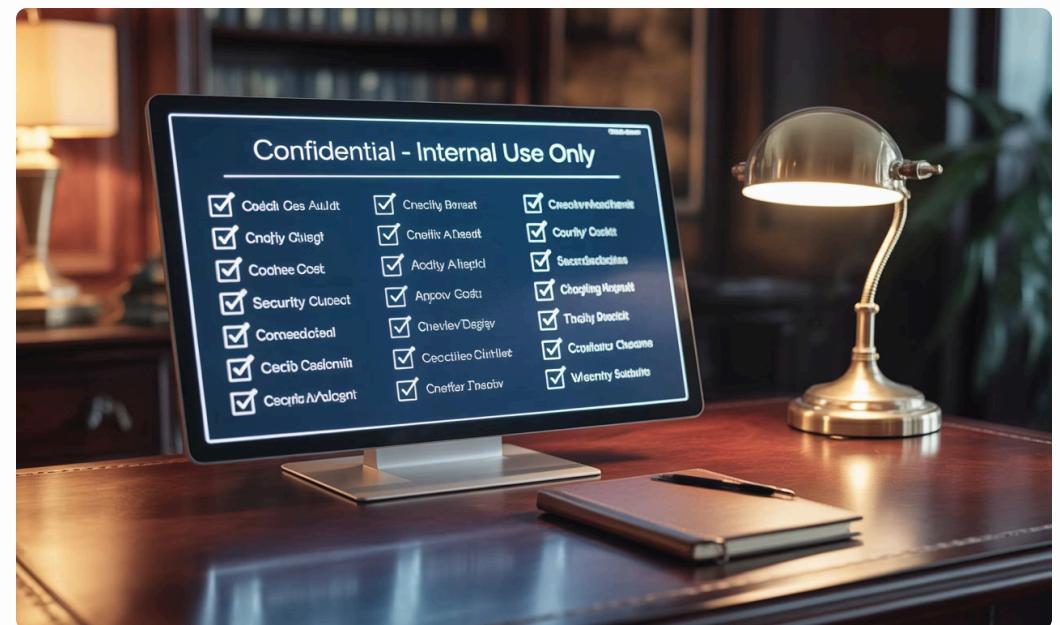
Organiser proprement et efficacement les tâches Ansible en créant un rôle.

Cela permet de :

- Réutiliser facilement du code.
 - Structurer clairement les fonctions (logs, durcissement, audit...).
 - Préparer un projet professionnel, lisible et modulaire.

Générer un rôle :

```
ansible-galaxy init roles/hardening_base
```



Copier les tâches dans

roles/hardening_base/tasks/main.yml

Crée un fichier

apply_hardening.yml

Structuration avec un rôle Ansible

Avantages de cette approche

Structure du rôle hardening_base

- Organisation modulaire des tâches
 - Réutilisation facile sur différents projets
 - Meilleure lisibilité et maintenance
 - Possibilité d'utiliser des dépendances entre rôles

Structure obtenue

```
roles/
└── hardening_base/
    ├── defaults/
    ├── files/
    ├── handlers/
    ├── meta/
    ├── tasks/
    ├── templates/
    ├── tests/
    └── vars/
```

```
dome@Ansible-Master:~/playbook$ sudo mkdir roles
dome@Ansible-Master:~/playbook$ ls -ld roles
drwxr-xr-x 2 root root 4096 24 juil. 10:40 roles

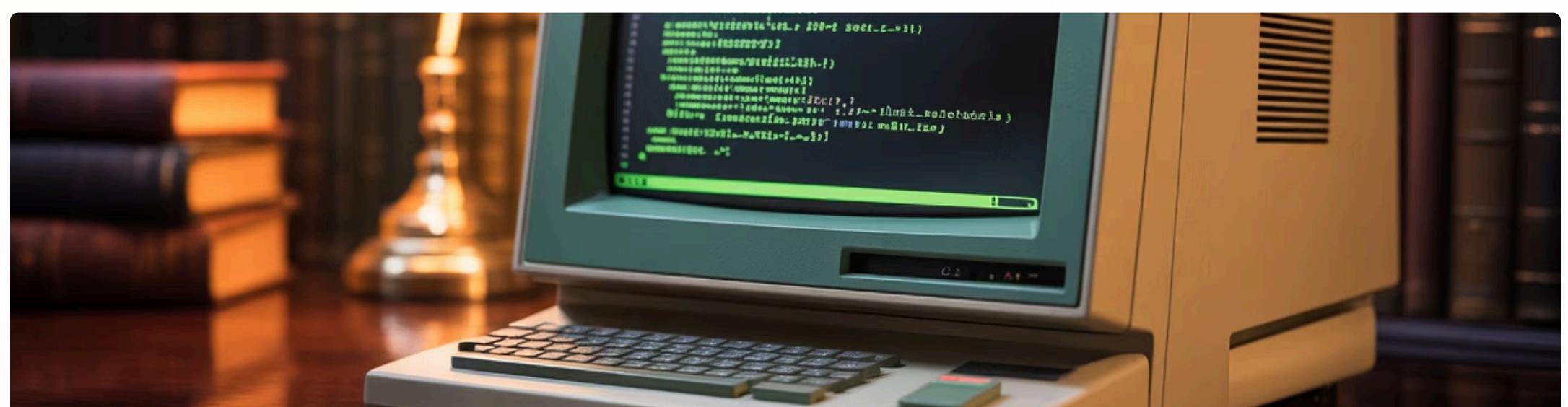
dome@Ansible-Master:~/playbook$ sudo chown -R dome:dome roles
dome@Ansible-Master:~/playbook$ ansible-galaxy init roles/hardening_base
- Role roles/hardening_base was created successfully
dome@Ansible-Master:~/playbook$ |
```

Copier les tâches dans `roles/hardening_base/tasks/main.yml`

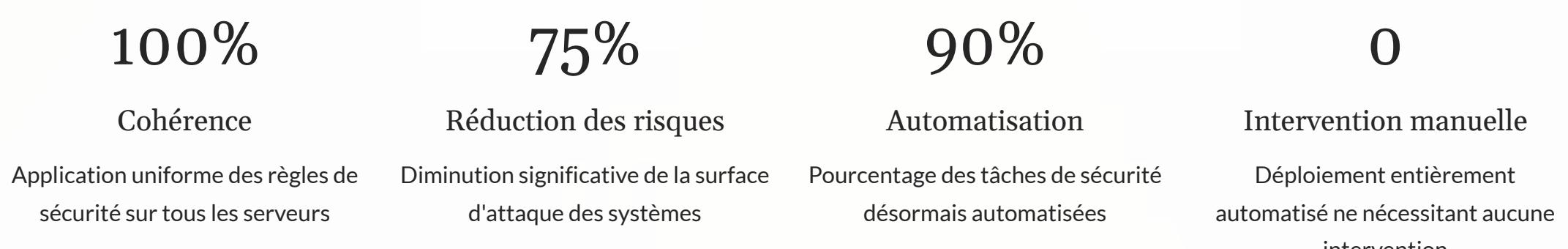
Créer un playbook pour utiliser ce rôle `apply_hardening.yml`

```
--  
- hosts: all  
  become: true  
  roles:  
    - hardening base
```

```
playbook/
├── inventory.ini
├── apply_hardening.yml
└── roles/
    └── hardening_base/
        ├── tasks/
        │   └── main.yml
        ├── defaults/
        │   └── main.yml
        ├── templates/
        └── ...
```



Résultat de la mise en place du rôle de durcissement



Jog 04 - Détection, Réponse à Incident et Cas d'Usage Avancés

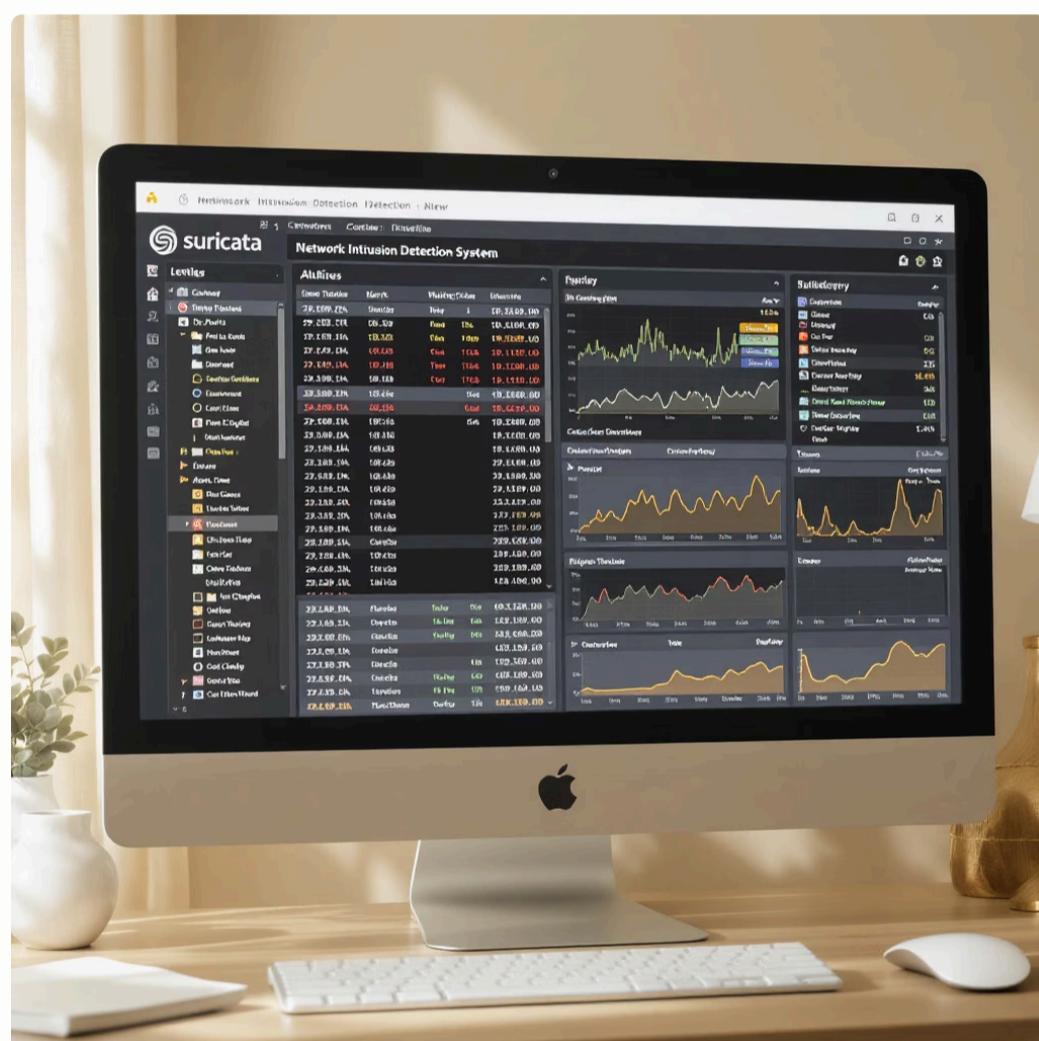
Objectif

Ce Job a pour objectif de simuler des mécanismes de détection d'intrusion, de réponse automatisée à un incident, de gestion sécurisée de secrets et d'extraction d'informations système via Ansible. Il initie à des usages avancés de la cybersécurité opérationnelle.



Déploiement d'un IDS/IPS simple (Suricata)

Ce module vise à installer un système de détection d'intrusion réseau (IDS) sur des machines cibles, configuré pour surveiller activement une interface réseau spécifique.



Étapes clés du déploiement



Choix de l'Outil

Utilisation de Suricata, une solution open-source robuste.



Installation du Paquet

Déploiement via le gestionnaire de paquets du système.



Configuration

Paramétrage minimal pour l'écoute sur l'interface réseau (ex: eth0).



Activation du Service

Mise en marche et démarrage automatique de Suricata.

Playbook Ansible : deploy_suricata.yml

```
- name: Déploiement de Suricata IDS
  hosts: all
  become: true
```

tasks:

```
- name: Installer Suricata
  apt:
    name: suricata
    state: present
    update_cache: yes
```

```
- name: Configurer Suricata pour écouter sur eth0
  lineinfile:
    path: /etc/suricata/suricata.yaml
    regexp: '^ *interface:'
    line: ' interface: eth0'
    insertafter: '^ *af-packet:'
```

```
- name: Redémarrer Suricata
  service:
    name: suricata
    state: restarted
    enabled: true
```

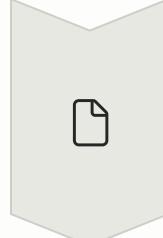
Vérification du service

```
sudo systemctl status suricata
```

Réponse à incident simulée

Objectif pédagogique

Simuler la détection d'un fichier malveillant et automatiser une réponse sécurisée et maîtrisée.



Création

Générer manuellement un fichier suspect pour simuler une intrusion (`/tmp/malicious_script.sh`).



Détection

Utiliser Ansible pour identifier la présence de ce fichier sur les systèmes cibles.

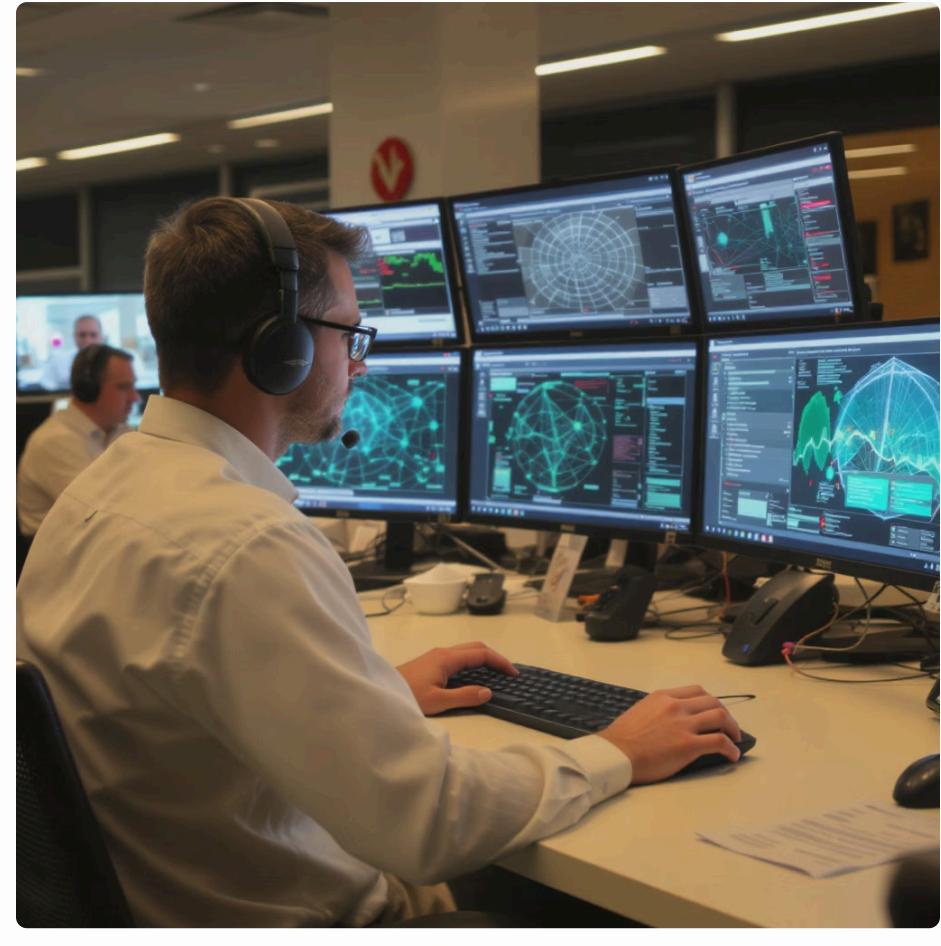


Réaction automatisée

Orchestrer une série d'actions sécurisées en réponse à la détection du fichier malveillant.

- **Arrêt d'un service compromis** (ex: nginx).
- **Mise en quarantaine** du fichier vers `/root/quarantine/`.
- **Collecte de données forensics** (logs et processus).
- **Notification d'incident** via un log sur le contrôleur.

Playbook Ansible : `incident_response.yml`



Vérification du déploiement

- Fichier déplacé dans `/root/quarantine` sur la machine ciblée.
- Log d'incident créé sur le contrôleur Ansible dans `./incident_logs/`.

```
---
- name: Réponse à incident simulée
  hosts: all
  become: true
  vars:
    suspicious_file: "/tmp/malicious_script.sh"
    quarantine_dir: "/root/quarantine"

  tasks:
    - name: Vérifier la présence du fichier suspect
      stat:
        path: "{{ suspicious_file }}"
      register: file_stat

    - name: Arrêter nginx si fichier suspect détecté
      service:
        name: nginx
        state: stopped
      when: file_stat.stat.exists

    - name: Créer le répertoire de quarantaine
      file:
        path: "{{ quarantine_dir }}"
        state: directory
        mode: '0700'
      when: file_stat.stat.exists

    - name: Déplacer le fichier vers la quarantaine
      command: mv {{ suspicious_file }} {{ quarantine_dir }}/
      when: file_stat.stat.exists

    - name: Collecter les processus actifs
      shell: ps aux
      register: process_list
      when: file_stat.stat.exists

    - name: Collecter les derniers logs
      shell: journalctl -n 50
      register: log_output
      when: file_stat.stat.exists

    - name: Écrire un log local sur le contrôleur
      delegate_to: localhost
      copy:
        content: |
          Incident détecté !
          --- Processus :
          {{ process_list.stdout }}

        --- Logs :
        {{ log_output.stdout }}
      dest: "./incident_logs/{{ inventory_hostname }}.log"
      when: file_stat.stat.exists
```

Utilisation d'Ansible Vault

Objectif

Protéger des données sensibles comme les clés API et les mots de passe directement dans vos playbooks Ansible, assurant ainsi la sécurité de vos configurations.



Création d'un secret avec Ansible Vault

```
ansible-vault create secrets.yml
```

Contenu de secrets.yml (chiffré):

```
api_key: "super-secret-key-123"
```

Utilisation dans un playbook (example_vault.yml)

```
---
- name: Exemple Vault
  hosts: localhost
  vars_files:
    - secrets.yml

  tasks:
    - name: Afficher une clé API chiffrée
      debug:
        msg: "La clé est : {{ api_key }}"
```

Commande d'exécution

```
ansible-playbook example_vault.yml --ask-vault-pass
```

Étapes Clés



Création

Générer un fichier chiffré via la commande Ansible Vault.



Intégration

Référencer ce fichier chiffré dans vos playbooks.



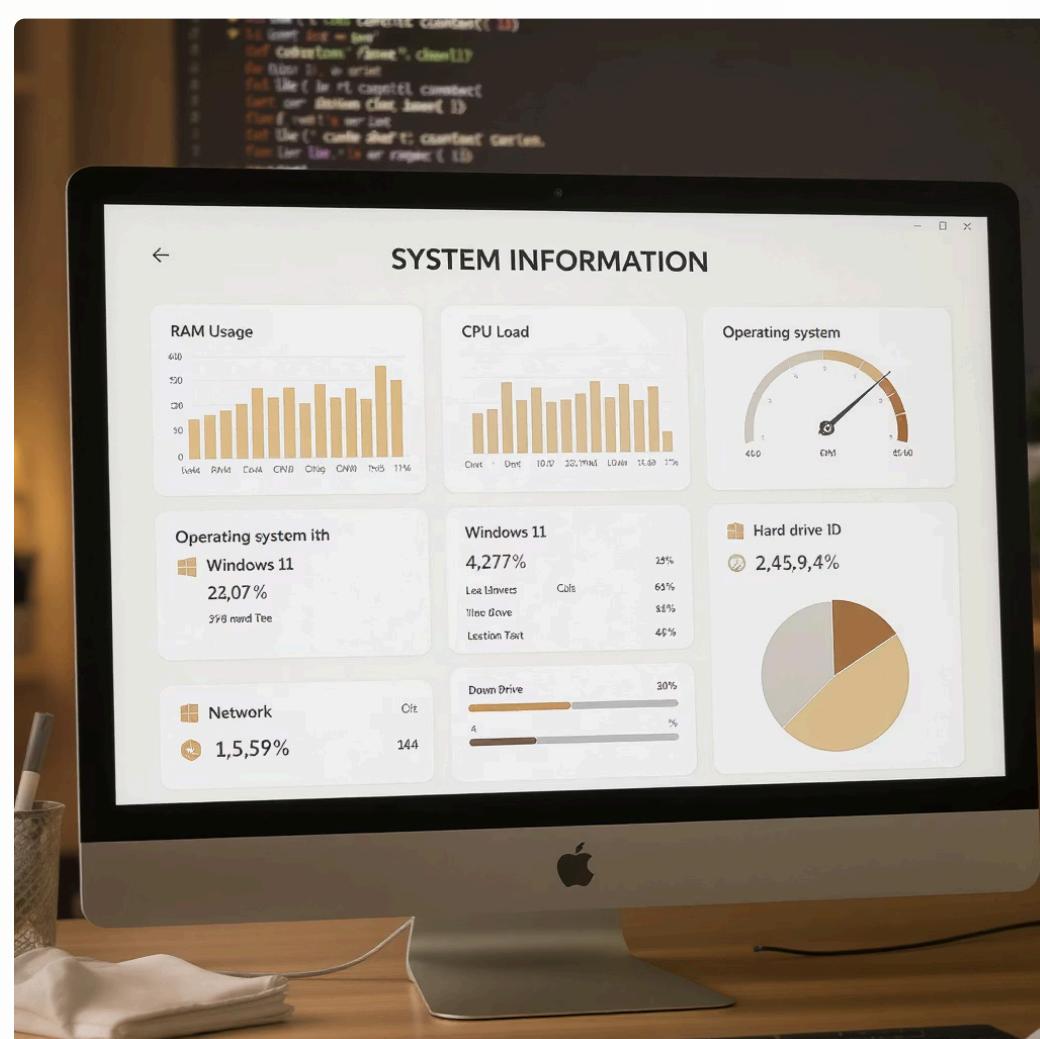
Exécution

Lancer le playbook en fournissant le mot de passe du coffre.

Gestion des Facts

Objectif

Collecter automatiquement des informations système (RAM, OS, etc.) détaillées à partir de vos hôtes gérés par Ansible. Ces informations sont essentielles pour l'inventaire, le diagnostic et l'adaptation des configurations, permettant une gestion plus précise et des déploiements conditionnels.



Exemple de Playbook : gather_facts.yml

```
---
- name: Affichage des facts
  hosts: all
  become: false

  tasks:
    - name: Récupérer les facts
      ansible.builtin.setup:

    - name: Afficher la RAM et OS
      ansible.builtin.debug:
        msg: "OS: {{ ansible_distribution }} {{ ansible_distribution_version }} - RAM: {{ ansible_memtotal_mb }} MB"
```

Ce playbook simple utilise le module `setup` pour collecter une multitude d'informations (facts) sur les systèmes cibles, puis le module `debug` pour afficher des détails spécifiques comme le système d'exploitation et la mémoire RAM disponible. Ces faits sont automatiquement collectés au début de chaque exécution de playbook, sauf si désactivé.

Job 5 – Scénario, Optimisation et Bilan

Ce dernier job a pour but de consolider toutes les compétences acquises en Ansible, depuis la sécurisation des serveurs jusqu'au déploiement complet d'une application web simple. Il vise également à structurer proprement le projet, optimiser les playbooks et produire un rapport final exploitable.

Déploiement sécurisé d'une application web avec Nginx

Déployer une page web HTML statique sur un serveur Linux en respectant des normes de sécurité strictes :

Serveur Durci

SSH, pare-feu, et autres configurations de sécurité de base.

Nginx HTTPS

Installation du serveur Nginx avec configuration HTTPS (certificat auto-signé).

Logging Filebeat

Intégration et configuration de Filebeat pour la collecte des logs.

Création du rôle webserver_secure

Organiser le rôle avec la structure suivante :

```
roles/
└── webserver_secure/
    ├── tasks/
    │   └── main.yml
    ├── templates/
    │   ├── nginx.conf.j2
    │   └── index.html.j2
    ├── files/
    │   └── certs/ (certificat + clé auto-signée)
    └── handlers/
        └── main.yml
```

Tâches à inclure dans le rôle (roles/webserver_secure/tasks/main.yml)

```
---
- name: Appliquer les rôles de hardening
  import_role:
    name: hardening_base

- name: Installer Nginx
  apt:
    name: nginx
    state: present
    become: yes

- name: Copier la page HTML statique
  template:
    src: index.html.j2
    dest: /var/www/html/index.html
    mode: '0644'

- name: Copier la configuration Nginx sécurisée
  template:
    src: nginx.conf.j2
    dest: /etc/nginx/sites-available/default
  notify: Redémarrer Nginx

- name: Copier le certificat SSL
  copy:
    src: certs/
    dest: /etc/nginx/certs/
    mode: '0644'
    owner: root
    group: root
  notify: Redémarrer Nginx

- name: S'assurer que Filebeat collecte les logs Nginx
  lineinfile:
    path: /etc/filebeat/filebeat.yml
    line: "- /var/log/nginx/*.log"
    insertafter: "^paths:"
  notify: Redémarrer Filebeat
```

Handlers (roles/webserver_secure/handlers/main.yml)

```
---
- name: Redémarrer Nginx
  service:
    name: nginx
    state: restarted

- name: Redémarrer Filebeat
  service:
    name: filebeat
    state: restarted
```

Vérifications attendues

- Accès à https://IP_serveur depuis un navigateur sans erreur de chargement (accepter le certificat auto-signé).
- curl -k https://localhost renvoie la page HTML.
- journalctl -u nginx ne montre aucune erreur.
- Filebeat tourne et les logs Nginx sont collectés.

Optimisation des playbooks

L'objectif principal de cette phase est de rendre le projet Ansible plus **maintenable, modulaire et lisible**. En adoptant les meilleures pratiques, nous assurons une gestion plus efficace et une meilleure collaboration au sein de l'équipe.

Utilisation de Rôles	Blocs de Tâches	Centralisation des Variables
Structurer les playbooks en rôles réutilisables.	Grouper les tâches logiquement pour une meilleure lisibilité.	Définir les variables clés dans <code>group_vars</code> pour une gestion centralisée.

Templates Jinja2	Gestion des Handlers
Dynamiser les fichiers de configuration avec des templates.	Déclencher des actions uniquement lorsque des changements surviennent.



Mise en œuvre concrète de l'optimisation

Création des variables personnalisées

Définition des variables spécifiques au rôle `webserver_secure` pour une configuration flexible de Nginx.

```
roles/webserver_secure/vars/main.yml
```

```
---
```

```
nginx_listen_port: 443
ssl_cert_path: /etc/nginx/certs/cert.pem
ssl_key_path: /etc/nginx/certs/key.pem
web_root: /var/www/html
```

Création d'un template de configuration Nginx

Utilisation de Jinja2 pour un fichier de configuration Nginx dynamique.

```
mkdir -p roles/webserver_secure/templates
```

```
nano roles/webserver_secure/templates/nginx.conf.j2
```

```
server {
    listen {{ nginx_listen_port }} ssl;
    server_name localhost;

    ssl_certificate {{ ssl_cert_path }};
    ssl_certificate_key {{ ssl_key_path }};

    root {{ web_root }};
    index index.html;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

Mise à jour des tâches du rôle

Intégration des variables et du template Nginx dans les tâches du rôle `webserver_secure`.

```
roles/webserver_secure/tasks/main.yml
```

```
---
```

```
- name: Installer nginx
apt:
  name: nginx
  state: present
  update_cache: yes
become: yes

- name: Créer le dossier de certificats
file:
  path: /etc/nginx/certs
  state: directory
  mode: '0755'

- name: Copier les certificats auto-signés
copy:
  src: certs/
  dest: /etc/nginx/certs/
  mode: '0644'
  notify: Redémarrer nginx

- name: Déployer la page HTML
template:
  src: index.html.j2
  dest: "{{ web_root }}/index.html"
  mode: '0644'
  notify: Redémarrer nginx

- name: Déployer la config nginx
template:
  src: nginx.conf.j2
  dest: /etc/nginx/sites-available/default
  mode: '0644'
  notify: Redémarrer nginx

- name: Activer nginx
service:
  name: nginx
  state: started
  enabled: true
```

Résumé global des bonnes pratiques d'optimisation

Utiliser des rôles	ansible-galaxy init roles/mon_role	roles/
Blocs (block)	Grouper plusieurs tâches conditionnelles	tasks/main.yml
Centraliser dans <code>group_vars</code>	Variables pour groupes d'hôtes	group_vars/webservers.yml
Templates Jinja2	Fichiers de config dynamiques	templates/ (ex: nginx.conf.j2)
Handlers	Redémarrer un service après changement (notify:)	handlers/main.yml

```
- block:
  - name: Installer les paquets nécessaires
  apt:
    name: "{{ item }}"
    state: present
  loop:
    - nginx
    - filebeat
when: ansible_os_family == "Debian"
```

Ce bloc garantit que l'installation de Nginx et Filebeat se fait uniquement sur les systèmes basés sur Debian, améliorant la portabilité de votre playbook.

Template **Jinja2** pour la configuration Nginx

```
server {
```

```
  listen 443 ssl;
```

```
  server_name localhost;
```

```
  ssl_certificate /etc/nginx/certs/cert.pem;
```

```
  ssl_certificate_key /etc/nginx/certs/key.pem;
```

```
  root /var/www/html;
```

```
  index index.html;
```

```
}
```

Ce template permet de générer dynamiquement le fichier de configuration Nginx, incluant les chemins pour les certificats SSL, qui peuvent être des variables Ansible.

Rapport Final et Soutenance



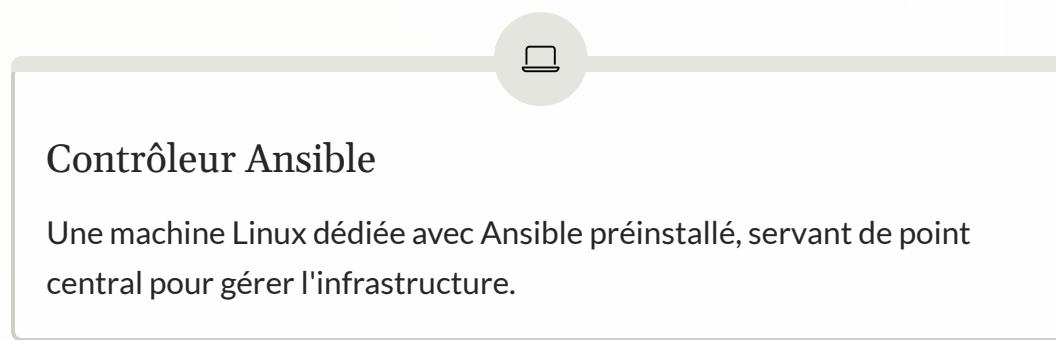
Introduction : L'Automatisation au Service de la Cybersécurité

Dans un contexte où les menaces numériques sont omniprésentes et évolutives, l'automatisation joue un rôle fondamental dans la sécurisation des systèmes d'information. Elle permet de déployer rapidement des correctifs, de durcir les configurations, de détecter des anomalies et de réagir aux incidents de manière uniforme et reproductible.

Ansible, en tant qu'outil d'automatisation sans agent, s'inscrit comme une solution de choix pour la cybersécurité. Il permet de gérer efficacement les infrastructures en réduisant les erreurs humaines et en assurant une traçabilité des actions, tout en facilitant la conformité aux politiques de sécurité.



Description de l'Environnement de TP



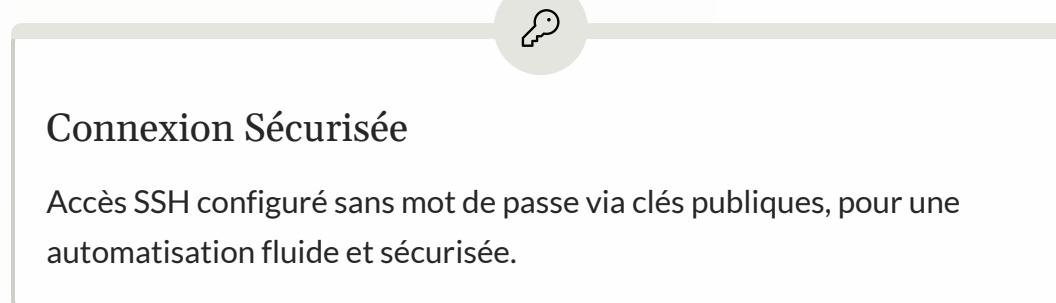
Contrôleur Ansible

Une machine Linux dédiée avec Ansible préinstallé, servant de point central pour gérer l'infrastructure.



Machines Cibles

Plusieurs machines Debian/Ubuntu et CentOS/Rocky Linux, représentant des environnements réels.



Connexion Sécurisée

Accès SSH configuré sans mot de passe via clés publiques, pour une automatisation fluide et sécurisée.



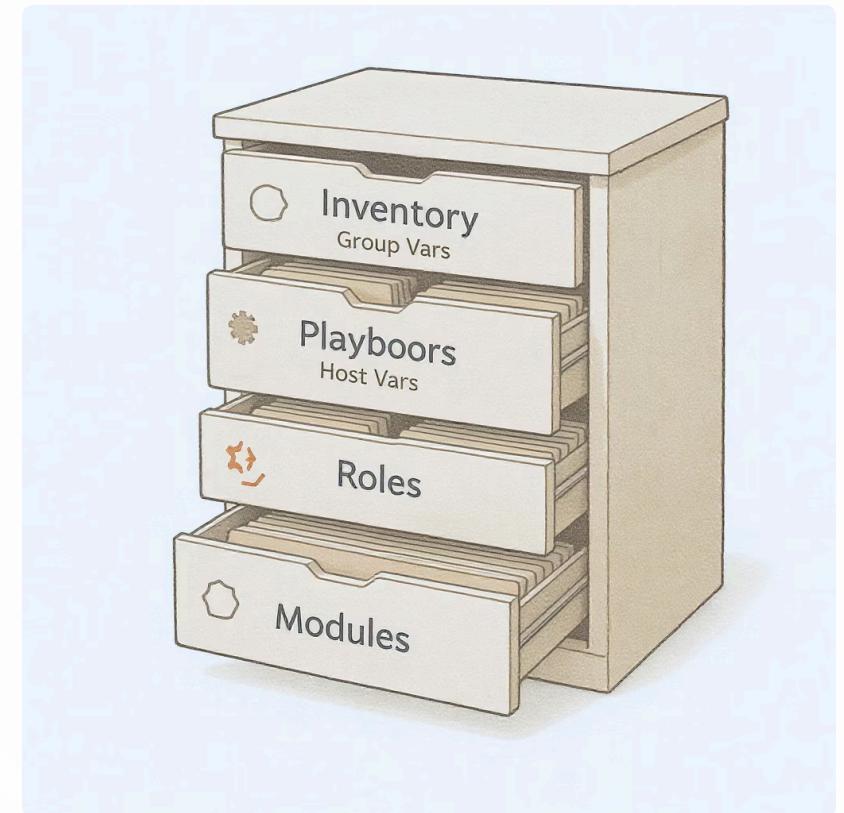
Inventaire Structuré

Organisation des hôtes en groupes logiques (webservers, dbservers, all_servers) pour une gestion précise.

Architecture Ansible Utilisée

L'organisation de ce projet Ansible suit les meilleures pratiques de structuration par rôles, assurant modularité et réutilisabilité. Voici l'architecture clé utilisée :

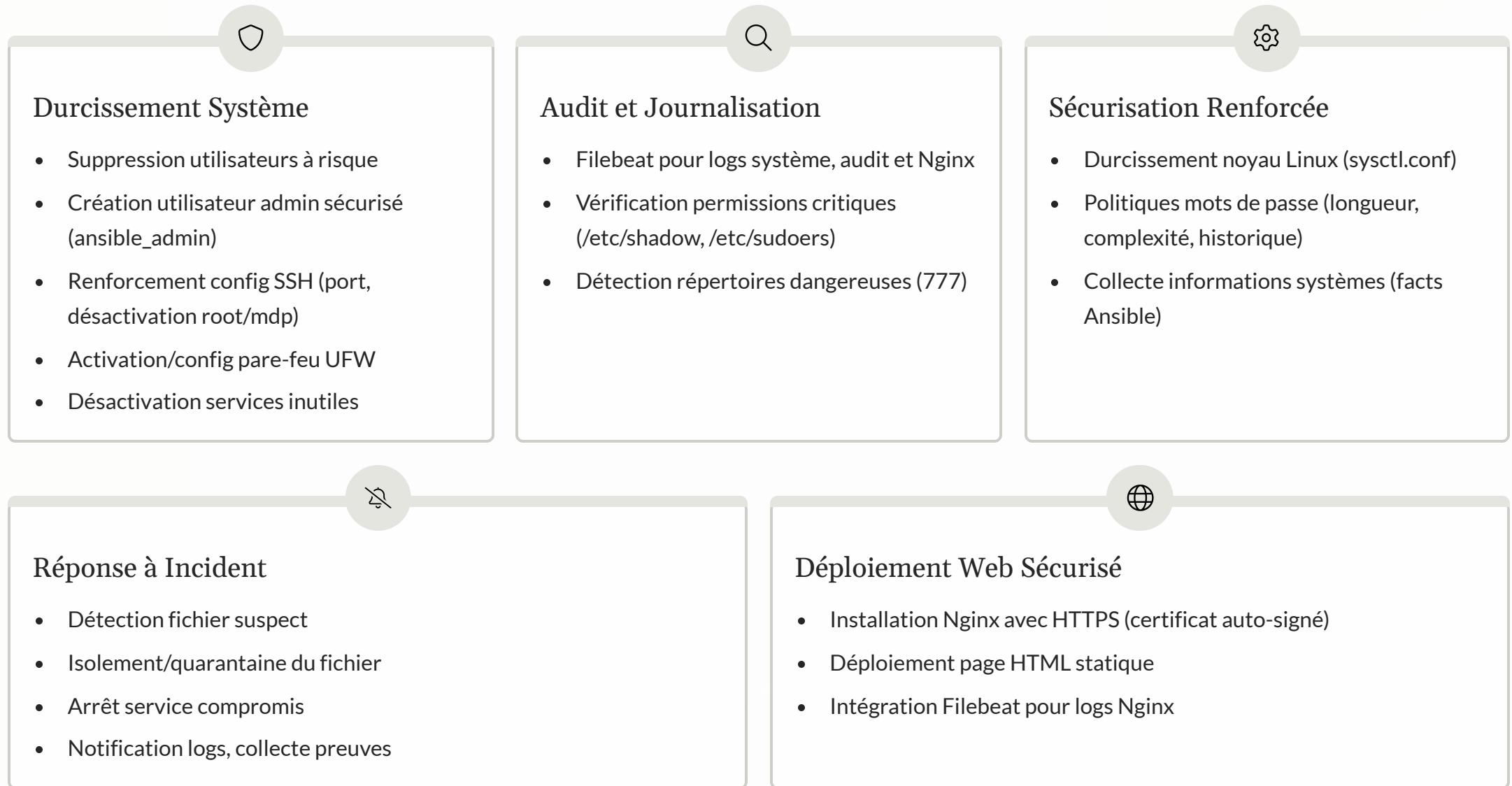
```
ansible/
├── inventory.ini
├── ansible.cfg
└── playbooks/
    ├── hardening.yml
    ├── filebeat.yml
    ├── web_deploy.yml
    └── incident_response.yml
├── roles/
    ├── hardening_base/
    ├── filebeat/
    ├── nginx_web/
    └── ids_response/
└── vault/
    └── secrets.yml (chiffré)
```



Détail des composants :

- **Inventaire** : Regroupement logique des machines cibles par usage.
- **Playbooks** : Fichiers YAML orchestrant des actions spécifiques (durcissement, logs, réponse à incident).
- **Rôles** : Encapsulation modulaire des tâches et configurations, facilement réutilisables.
- **Vault** : Protection des variables sensibles par chiffrement, garantissant la sécurité des informations confidentielles.

Mesures de Sécurité Implémentées



Conclusion : Efficacité d'Ansible pour la Cybersécurité

Le projet a démontré qu'Ansible est un outil puissant pour automatiser la sécurisation d'une infrastructure :



En conclusion, Ansible permet non seulement de gagner du temps mais aussi d'élever le niveau global de sécurité, même pour des équipes disposant de peu de ressources ou en phase d'apprentissage.