RÉPUBLIQUE DU BÉNIN
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY-CALAVI

INSTITUT DE FORMATION ET DE RECHERCHE EN INFORMATIQUE

BP 526 Cotonou Tel : +229 21 14 19 88
http://www.ifri-uac.bj Courriel : contact@ifri.uac.bj

MÉMOIRE

pour l'obtention du

Diplôme de Licence en Informatique

Option : Sécurité Informatique

Présenté par :
Igor Berenger **LOKOSSI**

Déploiement automatique d'un système d'IDS basé sur fail2ban

Sous la supervision :
Ingénieur Victor OYETOLA

Membres du jury :

SAHO Nelson	Docteur	Président
GOUDALO Jospy	Ingénieur	Examinateur
OYETOLA Victor	Ingénieur	Rapporteur

Année Académique : **2023-2024**

Sommaire

Dédicace ii

Remerciements iii

Résumé iv

Abstract v

Liste des figures vi

Liste des acronymes 1

Introduction 2

1 Revue de littérature 4

2 Notions, outils utilisés et déploiement 14

3 Évaluation et optimisation du système déployé 45

Conclusion 52

Webographie 53

Annexe 54

Table des matières 59

Dédicace

Je dédie ce mémoire à mes chers parents dont le soutien constant et les encouragements inébranlables ont été des piliers essentiels tout au long de mon parcours.

Remerciements

Je tiens à remercier chaleureusement tous ceux qui ont contribué à la réalisation de ce travail, en particulier :

- Professeur Eugène C. EZIN, Directeur de l'Institut de Formation et de Recherche en Informatique (IFRI) ;
- Monsieur Victor OYETOLA, mon maître de mémoire pour sa disponibilité et son soutien ;
- Tous les professeurs de l'Institut de Formation et de Recherche en Informatique (IFRI) pour l'enseignement qu'ils ont dispensé ;
- Ma famille pour tout le soutien qu'elle m'a apporté durant ces trois années de formation ;

Résumé

Ce mémoire présente un projet de déploiement automatique d'un système de détection d'intrusions basé sur Fail2ban, géré à l'aide d'Ansible. Le premier chapitre explore l'évolution des systèmes IDS, en soulignant les défis modernes auxquels les infrastructures sont confrontées face aux cybermenaces. Le deuxième chapitre détaille le processus pratique de configuration et de mise en place de Fail2ban, en insistant sur la personnalisation des configurations selon l'environnement cible. Enfin, le dernier chapitre évalue l'efficacité du système mis en place à travers des tests et présente des pistes d'amélioration pour de futures automatisations.

Mots clés : IDS (Intrusion Detection System), fail2ban, ansible, automatisation, analyse de logs.

Abstract

This thesis presents a project for the automatic deployment of an intrusion detection system based on Fail2ban, managed using Ansible. The first chapter explores the evolution of IDS systems, highlighting the modern challenges that infrastructures face in the face of cyber threats. The second chapter details the practical process of configuring and implementing Fail2ban, emphasizing the customization of configurations according to the target environment. Finally, the last chapter evaluates the effectiveness of the implemented system through tests and presents avenues for improvement for future automation.

Key words: IDS (Intrusion Detection System), fail2ban, ansible, automation, log analysis.

Liste des figures

2.1 Playbook d'installation de fail2ban . . . 18
2.2 Génération de clés ssh . . . 20
2.3 Ajout de la clé publique sur les nodes . . . 21
2.4 Arborescence du playbook . . . 23
2.5 Fichier inventaire . . . 24
2.6 Déclaration des variables globales . . . 25
2.7 Importation des tâches . . . 26
2.8 Playbook d'installation fail2ban . . . 28
2.9 Playbook de configuration fail2ban . . . 30
2.10 Include vars . . . 31
2.11 Fail2ban.local.j2 . . . 33
2.12 sshd.config.j2 . . . 34
2.13 ssh_jail.local.j2 . . . 35
2.14 pam_jail.local.j2 . . . 36
2.15 Fichier inventaire . . . 38
2.16 Déploiement du palybook . . . 38
2.17 Déploiement effectuée . . . 39
2.18 Ip machine test2 . . . 40
2.19 Ip machine test3 . . . 40
2.20 Service fail2ban test2 . . . 40
2.21 Service fail2ban test3 . . . 40
2.24 Clé ssh . . . 41
2.22 Service ssh test2 . . . 41
2.23 Service ssh test3 . . . 41
2.25 Copie cle ssh . . . 41
2.26 Inventaire . . . 42
2.27 Ping ssh . . . 42
2.28 Python test2 . . . 43
2.29 Python test3 . . . 43
2.30 Deploiement . . . 43
2.31 Fin deploiement . . . 43
2.32 Fail2ban machine test2 . . . 44
2.33 Fail2ban machine test3 . . . 44

3.1 Vérification de l'état de fail2ban . . . 46
3.2 Vérification paramètre de configuration sur le node . . . 46
3.3 État de bannissement fail2ban par défaut . . . 47

3.4 Tentative de connexion 47
3.5 Règle de bannissement définit pour ssh 48
3.6 Tentative de connexion après bannissement 48
3.7 Vérification de l'ip bannit 49
3.8 Fichier d'inventaire Ansible 54
3.9 Playbook d'installation de Fail2ban 55
3.10 Fail2ban.local.j2 55
3.11 ssh_jail.local.j2 56
3.12 Fail2ban machine test2 56
3.13 Fail2ban machine test3 57
3.14 Tentative de connexion après bannissement 57
3.15 Vérification de l'ip bannit 58

Liste des acronymes

AIDE : Advanced Intrusion Detection Environment

CPU : Central Processing Unit

DDoS : Distributed Denial of Service

HIDS : Host-based Intrusion Detection System

IDS : Intrusion Detection System
IP : Internet Protocol

NIDS : Network-based Intrusion Detection System

OS : Operating System
OSSEC : Open Source Security Information Management

RAM : Random Access Memory

SFTP : Secure File Transfer Protocol
SIEM : Security Information and Event Management
SSH : Secure Shell

TCP : Transmission Control Protocol

USM : Unified Security Management

YAML : Yet Another Markup Language

Introduction générale

Avec la croissance rapide des technologies numériques, les risques liés à la sécurité informatique augmentent chaque jour. Pour y faire face, des outils comme Fail2ban et Ansible sont devenus indispensables, car ils permettent d'automatiser la protection contre les tentatives d'intrusion, notamment les attaques par force brute. Ce mémoire présente notre travail sur le déploiement automatisé de Fail2ban via Ansible, en mettant en avant comment ces solutions peuvent aider à renforcer la sécurité des systèmes face aux menaces actuelles.

Contexte et justification

Avec l'évolution rapide des technologies de détection d'intrusions telles que Fail2ban et des outils d'automatisation comme Ansible, il est possible de relever efficacement les défis liés à la sécurité des infrastructures informatiques. Fail2ban se distingue par sa capacité à analyser en temps réel les journaux système pour bloquer automatiquement les adresses IP suspectes après un certain nombre de tentatives infructueuses. Associé à Ansible, il permet une gestion centralisée et cohérente de la sécurité, même dans des environnements complexes et hétérogènes.

Problématique

Face à l'augmentation des attaques automatisées, notamment les intrusions par force brute, la gestion manuelle des dispositifs de sécurité révèle ses limites. Il est donc crucial de déployer une solution IDS automatisée basée sur Fail2ban qui assure non seulement une détection rapide et une réaction adaptée en temps réel, mais aussi une gestion centralisée via Ansible, tout en minimisant les risques de faux positifs. Comment concevoir et automatiser un tel déploiement pour garantir une protection efficace et évolutive des infrastructures d'entreprise ?

Objectifs de l'étude

Cette étude vise à explorer en profondeur le déploiement automatisé d'un système d'IDS basé sur Fail2ban intégré avec Ansible, afin de renforcer la sécurité des infrastructures informatiques contre les menaces cybernétiques. Spécifiquement, les objectifs sont les suivants :

- Examiner les capacités de Fail2ban dans la détection en temps réel des tentatives d'intrusion et dans l'application automatisée de mesures de sécurité.
- Évaluer l'efficacité et l'impact de l'intégration d'Ansible pour le déploiement cohérent et la gestion centralisée des politiques de sécurité.
- Proposer des recommandations pratiques pour optimiser l'efficacité opérationnelle et la résilience des systèmes face aux attaques numériques.

Organisation du mémoire

Outre l'introduction et la conclusion, le présent mémoire est structuré en trois chapitres principaux. Le premier chapitre présente un état de l'art des systèmes IDS, en posant les bases avec les concepts essentiels de la sécurité informatique et en retraçant les évolutions majeures dans le domaine de la détection d'intrusions. Nous y abordons aussi les approches historiques et actuelles en sécurité des réseaux, ainsi que les principaux défis rencontrés par les IDS.
Le deuxième chapitre détaille notre méthode pour déployer automatiquement un système IDS basé sur Fail2ban, en utilisant Ansible. Nous expliquons la préparation des machines, l'installation et la configuration des outils, ainsi que la sécurisation des échanges et des accès. Ce chapitre introduit également les concepts de playbooks, de rôles et d'inventaires dans Ansible, essentiels pour le déploiement efficace et la gestion centralisée des politiques de sécurité.
Enfin, le troisième chapitre met en pratique les stratégies d'attaque et de défense que nous avons développées. Nous y présentons les résultats des tests fonctionnels, des simulations d'attaques, et les mesures de performance. Nous en tirons un bilan critique des points forts et des limites de notre solution, tout en proposant des pistes pour renforcer la sécurité des infrastructures face aux enjeux d'un environnement en perpétuelle évolution.

Chapitre 1

Revue de littérature

Introduction

La cybersécurité occupe aujourd'hui une place centrale dans la gestion des infrastructures informatiques. Pour renforcer efficacement la protection de nos systèmes, nous nous appuyons sur des outils tels que Fail2ban et Ansible, qui facilitent l'automatisation de la sécurité et la défense contre les tentatives d'intrusion. Dans ce chapitre, nous allons analyser en détail ces deux technologies, explorer leurs principales fonctionnalités et montrer comment elles contribuent à améliorer la protection de nos infrastructures.

1.1 Historique et évolution des IDS

1.1.1 Introduction aux Systèmes de Détection d'Intrusion (IDS)

Les systèmes de détection d'intrusion (IDS) jouent un rôle essentiel dans la protection des réseaux informatiques. Ils nous permettent d'identifier et de contrer les activités malveillantes afin de préserver l'intégrité, la confidentialité et la disponibilité de nos systèmes d'information. Historiquement, des outils comme Fail2ban et Snort ont marqué des étapes importantes dans ce domaine. Fail2ban bloque automatiquement les adresses IP suspectes en s'appuyant sur l'analyse des journaux système, tandis que Snort surveille le trafic réseau en temps réel pour identifier les menaces. Cependant, gérer ces outils manuellement s'avère rapidement complexe et chronophage, surtout face à l'évolution constante des menaces. L'automatisation du déploiement et de l'administration des IDS devient alors indispensable. Elle nous permet d'améliorer la réactivité, de renforcer la sécurité et d'alléger la charge de gestion. Dans cette partie, nous allons explorer ces enjeux et montrer les avantages d'un déploiement automatisé à travers l'exemple de Fail2ban.

1.1.2 Historique et développement des IDS

Avant l'apparition des IDS, la détection d'intrusions reposait entièrement sur des vérifications manuelles. Les administrateurs devaient analyser les journaux papier pour repérer d'éventuelles anomalies, ce qui était lent et peu efficace, car réalisé souvent après les attaques.

Dans les années 1970 et 1980, la baisse des coûts de stockage a permis de migrer ces journaux vers des serveurs numériques. Les premiers outils d'analyse de traces ont été développés, bien qu'ils soient encore lents et réservés aux périodes de faible activité.

En 1980, James Anderson a introduit le concept d'IDS. Quelques années plus tard, en 1987, Dorothy Denning a publié les premiers modèles de détection, posant les bases modernes des IDS.

Au début des années 1990, les premiers IDS capables d'analyser les journaux en temps réel sont apparus, permettant une détection plus rapide. À cette époque, les attaques étaient surtout menées par des experts. Avec la démocratisation d'outils de piratage, même des personnes peu qualifiées ont pu cibler des systèmes.

Entre 2006 et 2010, le nombre d'attaques a explosé, passant de 5 000 à plus de 35 000 par an. Cette augmentation a souligné l'urgence de développer des IDS plus performants face à la diversité croissante des cyberattaques.

Aujourd'hui, les IDS continuent d'évoluer pour sécuriser des environnements de plus en plus complexes. L'objectif reste de garantir une protection efficace malgré l'émergence constante de nouvelles menaces. [6].

1.2 Typologies des IDS

Les IDS peuvent être classés selon leur emplacement et leur méthode de détection.

Les **IDS basés sur l'hôte (HIDS)** ils sont installés directement sur les machines à protéger. Ils surveillent les journaux et les processus système. Ils sont efficaces contre des attaques ciblées sur un hôte (comme les rootkits).

Les **IDS basés sur le réseau (NIDS)** ils analysent le trafic réseau pour détecter des attaques comme les scans de ports ou les exploitations de vulnérabilités. Ils s'installent aux points stratégiques du réseau.

Les **IDS hybrides** ils combinent les deux approches précédentes pour une protection plus complète.

En fonction de la méthode de détection, les IDS peuvent être classés en **basés sur les signatures**, qui comparent les événements à une base de données de signatures connues d'attaques, ou **basés sur les anomalies**, qui détectent des comportements inhabituels ou suspects[6].

1.2.1 Comparaison avec d'autres outils de détection d'intrusions

Nous avons choisi Fail2ban pour cette étude, mais il existe plusieurs autres outils. Voici une comparaison simple et claire :

- **Suricata (NIDS, détection et prévention des intrusions)** :
  - Fail2ban se concentre sur la protection contre les tentatives de connexion brutales via des règles basées sur les journaux système, tandis que Suricata est un IDS en temps réel qui analyse le trafic réseau pour détecter des intrusions.
  - Suricata offre des capacités de prévention d'intrusion (IPS), ce qui lui permet de bloquer les attaques en temps réel, contrairement à Fail2ban qui agit uniquement en réponse à des comportements détectés dans les logs.

  - Suricata est beaucoup plus complexe à configurer et nécessite davantage de ressources, tandis que Fail2ban, par son approche simple, est plus léger et facile à déployer pour une protection basique contre les attaques.

- **Zeek (anciennement Bro) (NIDS, analyse approfondie du trafic)** :
  - Zeek se concentre sur l'analyse détaillée du trafic réseau, offrant une vue complète des communications réseau et des événements suspects. Fail2ban, en revanche, est limité à la surveillance des fichiers de log pour détecter des tentatives d'intrusion spécifiques.
  - Zeek fournit une surveillance proactive et une analyse des paquets en temps réel, mais il peut être plus complexe à mettre en place et à gérer. Fail2ban, bien que plus simple, ne fournit pas un niveau aussi approfondi de détection du trafic réseau.
  - En matière de prévention, Zeek peut être intégré à des systèmes comme Suricata pour une couverture complète, tandis que Fail2ban est généralement un complément à des outils de sécurité existants.

- **OSSEC (HIDS, détection basée sur les logs et les fichiers système)** :
  - OSSEC est un HIDS (Host Intrusion Detection System) qui surveille les logs et les fichiers système pour détecter des comportements suspects. Comme Fail2ban, OSSEC se concentre sur l'analyse des logs, mais OSSEC offre une couverture plus étendue avec une analyse comportementale et la possibilité de détecter des attaques internes.
  - OSSEC fournit également des fonctionnalités de gestion des configurations et de surveillance des systèmes, ce que Fail2ban ne propose pas. Cependant, Fail2ban est beaucoup plus simple à configurer et à utiliser.
  - OSSEC peut offrir une protection plus complète sur les hôtes, tandis que Fail2ban se limite à protéger les services d'accès, notamment contre les attaques par force brute.

- **Chkrootkit, Rkhunter, Tripwire (HIDS, détection de rootkits et de modifications système)** :
  - Chkrootkit, Rkhunter et Tripwire sont des outils spécifiques qui se concentrent sur la détection des rootkits et des modifications non autorisées du système. Fail2ban ne détecte pas ce type de menace, car il se concentre sur les attaques par force brute sur les services accessibles.
  - Ces outils sont plus spécialisés dans la détection de menaces internes, tandis que Fail2ban est conçu pour protéger contre des attaques externes telles que les tentatives de connexion non autorisées.
  - La combinaison de Fail2ban avec des outils comme Chkrootkit ou Tripwire peut offrir une protection plus complète contre une large gamme de menaces.

En résumé, Fail2ban se distingue par sa simplicité et son efficacité pour bloquer rapidement les attaques par force brute. Il est particulièrement utile comme première ligne de défense, et peut être combiné avec d'autres outils pour renforcer la sécurité.De plus, son intégration avec Ansible permet d'automatiser son déploiement ce qui est en ligne avec l'objectif de ce travail.

1.3 Contexte et choix de la solution

Dans le cadre de ce travail, nous avons choisi Fail2ban comme solution HIDS pour protéger les services exposés sur un serveur. Fail2ban analyse les journaux système et bloque automatiquement les adresses IP à l'origine d'activités suspectes.

L'un des défis majeurs réside dans la configuration manuelle de Fail2ban sur plusieurs machines. Pour simplifier cette tâche, nous proposons d'automatiser entièrement son déploiement et sa gestion avec Ansible. Cela nous permettra de :

- Déployer Fail2ban de façon centralisée sur plusieurs serveurs ;
- Assurer une configuration homogène ;
- Automatiser les mises à jour et les règles de sécurité.

Notre objectif est de fiabiliser la mise en place d'un IDS basé sur Fail2ban dans un environnement nécessitant une protection proactive.

1.4 Fail2ban : Présentation et fonctionnalités

1.4.1 Introduction à Fail2ban

Fail2ban est un outil clé dans la prévention des intrusions, notamment contre les attaques par force brute. En analysant les journaux système, il détecte les comportements suspects et bloque temporairement les adresses IP malveillantes. Il est flexible, adaptable, et son efficacité en fait un outil indispensable pour sécuriser des services exposés.

1.4.2 Fonctionnalités principales de Fail2ban

Fail2ban est un outil essentiel pour la protection des systèmes contre les attaques basées sur la force brute et d'autres comportements malveillants. Ses fonctionnalités clés sont les suivantes :

- **Analyse des fichiers de log en temps réel :** Fail2ban surveille les fichiers de log du système (ex. /var/log/auth.log) pour identifier des activités suspectes.Il détecte notamment les tentatives répétées de connexion infructueuses sur SSH, FTP, ou encore des attaques sur des services web.
- **Détection et réaction automatique aux menaces :** Une fois un comportement anormal détecté, Fail2ban applique une sanction préconfigurée.L'action la plus courante est le bannissement temporaire de l'IP via iptables, firewalld ou d'autres mécanismes.
- **Configuration flexible et personnalisable :** Les utilisateurs peuvent définir leurs propres règles (jail.conf, jail.local) pour adapter Fail2ban à leurs besoins spécifiques.
- **Génération de journaux et rapports d'activité :** Fail2ban fournit des logs détaillés sur les actions effectuées, facilitant ainsi la surveillance des menaces.Ces informations sont cruciales pour une analyse approfondie et une optimisation des stratégies de défense.

Nous avons choisi Fail2ban pour sa simplicité et son efficacité dans la protection des services critiques. Contrairement à Snort ou Suricata, qui analysent l'ensemble du trafic, Fail2ban se concentre sur la protection des accès aux services ce qui est particulièrement adapté à notre projet.

1.4.3 Avantages et limites de Fail2ban en tant qu'IDS

Dans le cadre de ce travail de déploiement automatisé d'un IDS basé sur Fail2ban, cet outil présente plusieurs avantages significatifs tout en ayant certaines limitations qu'il convient de prendre en compte pour optimiser sa mise en œuvre.

Avantages :

- **Protection contre les attaques par force brute :** Fail2ban se révèle particulièrement efficace pour contrer les attaques par force brute, qui figurent parmi les menaces les plus fréquentes sur les serveurs exposés à Internet. En surveillant les fichiers de journaux, il détecte automatiquement les tentatives de connexion infructueuses et bloque l'adresse IP responsable. Cette capacité est essentielle dans le cadre de notre travail, notamment pour sécuriser des services sensibles tels que SSH ou SFTP, où la moindre tentative d'accès non autorisé peut entraîner des conséquences importantes.

- **Adaptabilité aux différents services et protocoles :** Fail2ban peut être configuré pour surveiller un large éventail de services tels que SSH, SFTP, HTTP, entre autres. Cette flexibilité nous permet de l'adapter facilement à divers besoins de sécurité dans un environnement automatisé. Par exemple, nous pouvons l'utiliser pour protéger nos services web contre des attaques de type DDoS ou renforcer la sécurité des accès distants via SSH.

- **Réduction de la charge sur les services :** En bloquant automatiquement les adresses IP malveillantes, Fail2ban limite les tentatives de connexion infructueuses qui consomment inutilement les ressources du serveur. Dans un contexte automatisé, cette fonctionnalité optimise l'efficacité de nos services en permettant aux ressources système de se concentrer sur les utilisateurs légitimes.

- **Facilité de configuration et d'utilisation :**Bien que FMalgré sa flexibilité, Fail2ban reste simple à configurer. Grâce à ses fichiers de configuration (jail.conf, jail.local), nous pouvons automatiser la gestion des règles de filtrage et des actions. Cette simplicité facilite son intégration dans nos processus d'automatisation, ce qui garantit une protection continue sans intervention manuelle.

- **Notifications et rapports d'activité :** Fail2ban génère des journaux détaillés et peut nous alerter dès qu'une activité suspecte est détectée. Dans le cadre d'une solution automatisée, ces alertes peuvent être intégrées dans un processus de surveillance proactive, comme l'envoi d'e-mails ou l'affichage d'alertes sur un tableau de bord centralisé.

Limites :

- **Faux positifs :** L'un des principaux défis avec Fail2ban réside dans la gestion des faux positifs. Il peut arriver que des adresses IP légitimes soient bloquées par erreur, ce qui pourrait perturber le service. Pour limiter ce risque, nous devons configurer les filtres avec précision et réaliser des tests réguliers.

- **Efficacité limitée contre les attaques sophistiquées :** Fail2ban est très efficace contre les attaques simples, comme la force brute, mais il montre ses limites face à des attaques plus complexes (par exemple, celles menées via des botnets ou des adresses IP dynamiques). Pour une protection plus complète, il est recommandé de l'associer à d'autres outils comme Snort ou Suricata.
- **Dépendance aux journaux système :** Fail2ban s'appuie exclusivement sur les journaux système pour détecter les menaces. Si ces journaux sont mal configurés ou incomplets, certaines attaques pourraient passer inaperçues. Il est donc crucial de veiller à la bonne configuration et à la surveillance régulière des logs.
- **Complexité de la configuration des règles :** Bien que sa configuration de base soit simple, la personnalisation avancée des règles et des actions peut s'avérer complexe. Dans le cadre de ce projet de déploiement automatisé, nous devrons accorder une attention particulière à cette configuration pour éviter des erreurs susceptibles de compromettre la protection ou d'entraîner des interruptions de service.
- **Risque de déni de service (DoS) :** Si un grand nombre d'adresses IP sont bloquées simultanément, cela peut entraîner une surcharge des ressources système, voire un risque de déni de service. Il est donc nécessaire de paramétrer avec soin les seuils de bannissement afin de préserver la disponibilité des services.

En conclusion, bien que Fail2ban soit un outil puissant pour renforcer la sécurité de votre système, il est important de prendre en compte ces conséquences potentielles et de mettre en place des mesures appropriées pour atténuer les risques associés. Cela peut inclure la configuration prudente des filtres et des actions, la surveillance régulière du système et la mise en place de politiques de gestion des faux positifs[1].

1.5 Ansible : Présentation et fonctionnalités

1.5.1 Introduction à Ansible

Ansible est une plateforme open-source d'automatisation de la configuration, du déploiement et de la gestion des infrastructures informatiques. Grâce à des playbooks écrits en YAML, nous pouvons décrire l'état désiré de nos systèmes et laisser Ansible appliquer ces configurations de manière cohérente. Ansible est apprécié pour sa simplicité, son absence d'agent (il utilise SSH et Python déjà disponibles sur les machines) et sa capacité à automatiser une large variété de tâches.

1.5.2 Fonctionnalités principales d'Ansible

Ansible est un outil puissant utilisé pour l'automatisation et la gestion des configurations. Il repose sur une architecture sans agent, ce qui signifie qu'il ne nécessite pas l'installation d'un logiciel supplémentaire sur les hôtes cibles. Au lieu de cela, Ansible se connecte aux machines distantes via SSH, en utilisant des outils déjà disponibles sur la plupart des systèmes Linux, à savoir SSH et Python. Cela simplifie le processus de gestion des hôtes et permet une automatisation à grande échelle[5].

Voici quelques-unes des fonctionnalités principales d'Ansible :

- **Déploiement simple et agentless :** Ansible fonctionne sans agent sur les hôtes cibles, ce qui simplifie son déploiement et sa gestion. En utilisant uniquement SSH pour se connecter aux hôtes et exécuter les tâches définies, Ansible peut facilement être intégré dans des environnements hétérogènes, réduisant ainsi la complexité de son installation.
- **Infrastructure as Code (IaC) :** Ansible permet de décrire l'état désiré de l'infrastructure dans des fichiers YAML appelés "playbooks". Cette approche déclarative permet de déployer et de gérer l'infrastructure de manière cohérente et reproductible, garantissant ainsi une gestion fiable et une mise à jour facile des configurations.
- **Automatisation des tâches :** Ansible couvre une large gamme de tâches administratives, telles que la configuration des serveurs, le déploiement des applications, la gestion des mises à jour, etc. Grâce à ses modules, il offre une flexibilité permettant de répondre à divers besoins d'automatisation dans un environnement complexe.
- **Orchestration :** Ansible permet d'orchestrer des déploiements sur un grand nombre de serveurs, soit en parallèle, soit de manière séquentielle. Cela le rend particulièrement adapté pour déployer des applications complexes sur des infrastructures distribuées de manière efficace et fiable.
- **Gestion des configurations :** Ansible simplifie la gestion des configurations en permettant de définir et maintenir des configurations standardisées pour l'ensemble de l'infrastructure. Les playbooks Ansible peuvent être réutilisés et versionnés pour garantir la cohérence des configurations à travers tous les environnements.
- **Intégration avec d'autres outils :** Ansible s'intègre facilement avec d'autres outils de développement et d'administration système, comme Git pour le contrôle de version, Jenkins pour l'intégration continue, et des outils de surveillance et de gestion des incidents, permettant une automatisation fluide à travers l'ensemble du cycle de vie de l'application.

Ces fonctionnalités font d'Ansible un outil clé pour les administrateurs systèmes, les ingénieurs DevOps, et les développeurs, facilitant l'automatisation et la gestion de systèmes complexes tout en offrant une grande flexibilité.

1.5.3 Comparaison avec d'autres outils d'automatisation

Dans la sphère de l'automatisation des configurations et des déploiements, Ansible se distingue par plusieurs aspects qui le différencient des autres outils. Voici une comparaison avec quelques-uns de ses principaux concurrents :

- **Chef** :
  - Ansible utilise une approche agentless (ne nécessite pas l'installation d'un logiciel agent sur les dispositifs à gérer), tandis que Chef nécessite l'installation et la configuration d'un agent sur chaque hôte.
  - Chef utilise un modèle de configuration de type impératif, tandis qu'Ansible adopte une approche déclarative, permettant de décrire l'état désiré du système sans spécifier les étapes exactes pour y parvenir.

  - Ansible est souvent considéré comme plus facile à apprendre et à utiliser que Chef en raison de sa syntaxe simple et de sa courbe d'apprentissage moins raide.

- **Puppet** :
  - Tout comme Chef, Puppet utilise un modèle d'agent, tandis qu'Ansible est agentless.
  - Puppet est particulièrement adapté aux environnements où une grande précision dans la configuration est requise, tandis qu'Ansible brille dans les environnements nécessitant une automatisation rapide et flexible.
  - Ansible utilise YAML pour la définition des playbooks, ce qui rend sa syntaxe plus lisible et plus intuitive que celle de Puppet, qui repose sur un langage de programmation spécifique.

- **SaltStack** :
  - SaltStack est similaire à Ansible dans le sens où il peut être utilisé de manière agentless, mais il offre également des fonctionnalités d'agent pour les cas d'utilisation nécessitant une communication bidirectionnelle persistante.
  - SaltStack est souvent considéré comme plus complexe à configurer et à utiliser qu'Ansible, mais il offre des fonctionnalités avancées de gestion des configurations et de surveillance en temps réel[3].

En résumé, Ansible se distingue par sa simplicité d'utilisation, son approche agentless, et sa flexibilité dans la gestion des configurations et des déploiements. Bien qu'il existe d'autres outils d'automatisation sur le marché avec des fonctionnalités similaires, Ansible reste un choix populaire en raison de sa facilité d'apprentissage et de sa large adoption dans la communauté.

1.6 Déploiement automatique et sécurité informatique

1.6.1 Importance du déploiement automatique dans la sécurité informatique

Le déploiement automatique occupe une place stratégique dans le domaine de la sécurité informatique. En limitant les interventions manuelles, il réduit considérablement les risques d'erreurs humaines qui peuvent introduire des vulnérabilités. Grâce à la standardisation des processus, nous obtenons des configurations homogènes et fiables, ce qui renforce la solidité de notre infrastructure.

Cette automatisation nous permet également de réagir plus rapidement face aux menaces. Les correctifs de sécurité et les mises à jour peuvent être appliqués sans délai, ce qui diminue fortement la période durant laquelle nos systèmes restent exposés aux failles, même dans des environnements complexes et changeants.

L'assurance d'une configuration uniforme est un avantage majeur, en particulier lorsque nous gérons des infrastructures vastes ou en pleine expansion, comme dans les environnements cloud. Dans ces contextes, une gestion manuelle deviendrait vite inefficace.

Par ailleurs, l'automatisation facilite le suivi et la conformité. Toutes les actions de déploiement sont enregistrées, ce qui nous permet de disposer de traces précises pour les audits de sécurité et de garantir le respect des exigences réglementaires.

Enfin, en déchargeant nos équipes des tâches répétitives, le déploiement automatique leur permet de se concentrer sur des missions plus stratégiques, telles que la veille sur les menaces et la gestion des incidents. L'intégration d'outils comme Ansible nous offre ainsi une double opportunité : améliorer notre efficacité au quotidien et renforcer durablement la sécurité de nos systèmes.

1.7 Bénéfices attendus et défis

1.7.1 Bénéfices attendus

Le déploiement automatique des IDS apporte plusieurs avantages significatifs dans la gestion de la sécurité de nos systèmes. En automatisant les configurations et les déploiements, nous réduisons le risque d'erreurs humaines, ce qui limite les vulnérabilités liées aux manipulations manuelles. Cette approche nous permet d'améliorer la rapidité, l'efficacité et la cohérence de nos déploiements, en facilitant une réaction rapide face aux nouvelles menaces.

En appliquant des configurations de manière uniforme sur l'ensemble de l'infrastructure, nous renforçons la solidité de notre posture de sécurité. L'automatisation simplifie également la gestion d'environnements étendus, en nous donnant la capacité de déployer des configurations sur plusieurs centaines, voire milliers de serveurs en parallèle.

Par ailleurs, elle garantit une traçabilité complète de toutes les actions réalisées, un aspect essentiel pour les audits de sécurité et le respect des exigences réglementaires. Enfin, en allégeant les tâches répétitives, elle libère nos équipes afin qu'elles puissent se consacrer à des missions stratégiques telles que l'analyse des menaces et la gestion des incidents.

1.7.2 Défis

Malgré ses nombreux atouts, le déploiement automatique des IDS s'accompagne de plusieurs défis qu'il nous faut anticiper. La configuration initiale demande une préparation minutieuse et une planification rigoureuse. La moindre erreur dans les scripts ou les playbooks peut entraîner des configurations incorrectes, compromettant ainsi la sécurité du système. De plus, la gestion continue des mises à jour, des vulnérabilités et des évolutions de l'infrastructure reste un enjeu constant.

La gestion des alertes générées par les IDS, en particulier la maîtrise des faux positifs et des faux négatifs, nécessite un calibrage précis et une surveillance attentive. Nous devons également veiller à former nos équipes à l'utilisation de ces outils et à la gestion des configurations, ce qui représente une charge de travail supplémentaire à prendre en compte. Enfin, assurer une intégration fluide avec d'autres systèmes de sécurité déjà en place peut parfois s'avérer complexe. [2].

En résumé, même si l'automatisation des IDS offre des bénéfices indéniables, elle exige une planification soignée, des tests approfondis et une formation adéquate afin d'en maximiser l'efficacité tout en minimisant les risques.

Conclusion

Dans ce chapitre, nous avons souligné les enjeux liés à l'automatisation du déploiement des systèmes de détection d'intrusion pour renforcer la sécurité informatique. Nous avons détaillé les principales fonctionnalités de Fail2ban, en insistant sur ses points forts et ses limites, puis présenté Ansible comme outil d'automatisation, avec une comparaison à d'autres solutions. Enfin, nous avons rappelé l'importance de maintenir et adapter ces systèmes dans le temps. Le chapitre suivant portera sur la surveillance continue, la gestion des alertes et l'ajustement des configurations face aux menaces évolutives.

Chapitre 2

Notions, outils utilisés et déploiement

Introduction

Ce chapitre pose les fondations nécessaires à la mise en place d'un système de détection d'intrusion automatisé, en s'appuyant sur l'outil Fail2Ban et le gestionnaire de configuration Ansible. Nous y aborderons les notions essentielles de sécurité informatique, ainsi que le rôle stratégique de la détection des intrusions. Nous décrirons ensuite les étapes d'installation et de configuration des deux outils, avant de présenter la préparation de l'environnement de travail, incluant la configuration des machines et les mesures de sécurité. Enfin, nous expliquerons comment automatiser ce processus sur plusieurs hôtes grâce à la gestion d'inventaire d'Ansible. L'ensemble de ce chapitre vise à fournir une compréhension claire et structurée d'un déploiement IDS automatisé et évolutif.

2.1 Notions de base

2.1.1 Définition et importance de la sécurité informatique

La sécurité des systèmes d'information (SSI) regroupe un ensemble de mesures techniques, organisationnelles et humaines visant à protéger les ressources numériques contre l'accès non autorisé, la modification ou l'utilisation abusive. Elle permet d'assurer la protection du patrimoine numérique des organisations.

2.1.2 Principaux objectifs de la sécurité informatique

Les piliers fondamentaux de la sécurité des systèmes d'information peuvent être résumés à travers les critères suivants :

- **Confidentialité** : Assure que seules les personnes autorisées ont accès aux données sensibles. Cela repose sur des mécanismes de contrôle d'accès, de chiffrement et d'authentification.
- **Authenticité** : Garantit l'identification fiable des utilisateurs et des systèmes. Elle repose sur l'usage de mots de passe, de certificats numériques ou encore de l'authentification multi-facteurs.

- **Intégrité** : Veille à ce que les données ne soient pas altérées accidentellement ou malicieusement. L'intégrité est assurée par des mécanismes de signature numérique, hachage et contrôle de redondance.
- **Disponibilité** : Garantit l'accès aux ressources et services en temps voulu. Cela implique la mise en place de sauvegardes, de redondance et de mesures contre les attaques par déni de service (DDoS).

2.1.3 Autres aspects de la sécurité informatique

Outre les objectifs principaux, d'autres principes renforcent la sécurité des systèmes :

- **Traçabilité** : Permet d'enregistrer et d'analyser les actions effectuées sur le système. Elle repose sur la journalisation des événements et l'audit de sécurité.
- **Non-répudiation** : Empêche un utilisateur de nier une action qu'il a réalisée. Ce principe repose sur l'usage de signatures électroniques et de certificats.
- **Gestion des risques** : L'identification et l'analyse des menaces permettent de mettre en place des contre-mesures adaptées, réduisant ainsi l'impact potentiel d'une faille de sécurité[4].

La sécurité des systèmes repose sur une approche globale, prenant en compte aussi bien les vulnérabilités techniques que les erreurs humaines et les menaces externes. Une stratégie efficace implique une surveillance continue et une mise à jour régulière des mesures de protection.

2.1.3.1 Types de menaces et attaques courantes

La cybersécurité doit faire face à une diversité de menaces et d'attaques qui peuvent compromettre l'intégrité, la confidentialité et la disponibilité des systèmes informatiques. Voici les principales catégories d'attaques :

- **Attaque par force brute** : Tentative systématique de toutes les combinaisons possibles pour deviner un mot de passe ou une clé de chiffrement.
- **Attaques par déni de service (DDoS)** : Saturation d'un service ou d'un réseau en le submergeant de requêtes massives pour le rendre indisponible.
- **Logiciels malveillants (Malwares)** : Programmes conçus pour nuire aux systèmes, incluant virus, vers, chevaux de Troie, ransomwares et spywares.
- **Phishing et ingénierie sociale** : Tromperie visant à obtenir des informations sensibles via des e-mails frauduleux, des sites usurpés ou des manipulations psychologiques.
- **Attaques de type Man-in-the-Middle (MitM)** : Interception et modification des communications entre deux parties à leur insu, souvent via des réseaux non sécurisés.

Ces menaces soulignent la nécessité d'une protection proactive et de la mise en place de stratégies de défense adaptées aux risques.

2.1.4 Importance de la détection d'intrusions

2.1.4.1 Rôle de la détection d'intrusions dans la protection des systèmes

Les systèmes de détection d'intrusion (IDS) permettent de repérer rapidement des activités anormales ou malveillantes au sein d'un système informatique. En analysant les journaux, le trafic réseau ou d'autres sources, ils identifient soit des signatures d'attaques connues, soit des comportements inhabituels. Ces alertes permettent aux administrateurs de réagir avant qu'un dommage ne survienne. En ce sens, les IDS jouent un rôle essentiel dans la protection des ressources, en contribuant à maintenir la confidentialité, l'intégrité et la disponibilité des données.

2.1.4.2 Avantages de la détection proactive des intrusions

La détection proactive des intrusions offre des avantages significatifs, notamment la réduction des dommages potentiels, l'amélioration de la réactivité, la prévention des brèches de sécurité, l'enrichissement de l'intelligence de sécurité, la conformité réglementaire et l'amélioration de la posture de sécurité globale. Ces avantages sont essentiels pour protéger les systèmes informatiques et les données sensibles dans un environnement de menaces de plus en plus sophistiquées [7].

2.1.5 Concepts de base de Fail2Ban et Ansible

2.1.5.1 Fail2Ban

Fail2Ban est un outil de sécurité utilisé principalement sur les systèmes Linux pour empêcher les tentatives d'intrusion. Il agit en surveillant les journaux des services réseau afin de repérer les comportements suspects, comme des tentatives répétées de connexion échouées. Lorsqu'une adresse IP semble malveillante, Fail2Ban la bloque automatiquement en appliquant des règles au pare-feu. Ces blocages sont généralement temporaires et peuvent être configurés pour alerter l'administrateur. Cette approche permet de limiter les attaques par force brute et de renforcer efficacement la protection des serveurs.

2.1.5.2 Ansible

Ansible est une plateforme open-source populaire de gestion de la configuration, de déploiement d'applications et d'automatisation des tâches informatiques. Développé par Red Hat, Ansible utilise une approche déclarative pour décrire les états souhaités des systèmes informatiques, ce qui permet aux utilisateurs de décrire l'état souhaité d'un système dans un fichier texte simple, plutôt que de préciser étape par étape comment atteindre cet état. Cela le rend relativement facile à apprendre et à utiliser, même pour ceux qui ne sont pas familiers avec le langage de programmation ou les scripts complexes.

Ansible utilise SSH pour se connecter aux machines distantes et effectuer des opérations, ce qui signifie qu'il n'est pas nécessaire d'installer un agent sur les hôtes cibles, ce qui simplifie le processus de gestion. Il permet également une automatisation puissante et reproductible à grande échelle, ce qui en fait un outil précieux pour les administrateurs système, les ingénieurs DevOps et les développeurs. Ansible est un outil sans agent, ce qui signifie qu'il n'installe pas de logiciel supplémentaire sur les nœuds. Il fonctionne en mode push, envoyant les configurations directement aux nœuds. Pour ce faire, il se sert uniquement des outils déjà disponibles sur la plupart des systèmes Linux, à savoir SSH et Python.

Les principaux composants d'Ansible incluent les playbooks, les rôles et les inventaires. Les playbooks sont des fichiers écrits en YAML qui décrivent les tâches qu'Ansible doit exécuter sur les hôtes spécifiés. Ils constituent le cœur des opérations d'Ansible, permettant de définir les états désirés pour les systèmes gérés. Un playbook peut contenir une ou plusieurs "plays". Chaque play spécifie les hôtes cibles et les tâches à exécuter[5].

2.2 Présentation des outils utilisés

2.2.1 Ansible :

2.2.1.1 Installation et configuration de base

- **Procédure d'installation sur différents systèmes d'exploitation**

  Mettre à jour les paquets avec la commande «sudo apt-get update». Ensuite, avec la commande « sudo apt install ansible» lancer l'installation. Avec la commande «ansible –version» vérifier la version de Ansible installée sur la machine .

- **Configuration initiale et structure des fichiers de configuration**

  Après avoir installé Ansible, il est important de configurer correctement l'environnement pour pouvoir gérer efficacement les configurations et les déploiements. La configuration d'Ansible repose principalement sur quelques fichiers clés : ansible.cfg, le fichier d'inventaire, et les playbooks.

  Le fichier ansible.cfg permet de personnaliser le comportement d'Ansible. Vous pouvez le placer dans le répertoire de votre projet ou dans /etc/ansible/ pour une configuration globale [5].

2.2.1.2 Concepts de playbooks, rôles et inventaires :

Définition et Utilisation des Playbooks

Les playbooks sont des fichiers écrits en YAML qui décrivent les tâches qu'Ansible doit exécuter sur les hôtes spécifiés. Ils constituent la base des opérations d'Ansible et permettent de définir l'état souhaité pour les systèmes gérés.

Utilisation des playbooks :

- Définir les Tâches : Les playbooks contiennent une ou plusieurs "plays". Chaque play spécifie les hôtes cibles et les tâches à exécuter.
- Idempotence : Les tâches sont définies de manière à pouvoir être exécutées plusieurs fois sans provoquer de changements indésirables si l'état désiré est déjà atteint.
- Modules : Les tâches utilisent des modules Ansible pour effectuer des actions spécifiques, comme installer des paquets, copier des fichiers, ou gérer des services.
- Rôles : Les rôles permettent d'organiser les playbooks et les tâches de manière modulaire et réutilisable. Chaque rôle contient des sous-répertoires prédéfinis pour les tâches, les handlers, les fichiers, les templates et les variables.

- Structure des Rôles : Un rôle est organisé en plusieurs sous-répertoires, chacun ayant une fonction spécifique (tasks, handlers, files, templates, vars, defaults, meta).
- Réutilisabilité : Les rôles peuvent être utilisés dans plusieurs playbooks, facilitant ainsi la réutilisation des configurations et des tâches.

2.2.2 Fail2ban :

2.2.2.1 Installation et configuration de base

Procédure d'installation sur différents systèmes d'exploitation

Fail2Ban est un outil populaire utilisé pour protéger les systèmes contre les tentatives d'intrusion en bloquant temporairement les adresses IP suspectes. Son installation varie légèrement en fonction du système d'exploitation utilisé et aussi de l'outil utilise. Dans notre cas nous allons utiliser ansible donc les playbooks . La figure suivante illustre un exemple de playbook facilitant l'installation de Fail2Ban.

```
1  ---
2    - name: Install fail2ban
3      apt:
4        name: fail2ban
5        state: present
6
7    #- name: Install fail2ban-systemd
8    #  apt:
9    #    name: fail2ban-systemd
10   #  state: present
11
12   - name: Enable and Start fail2ban
13     systemd:
14       name: fail2ban
15       state: restarted
16       enabled: yes
17 ...
```

FIGURE 2.1 : Playbook d'installation de fail2ban

Observation :

Cette tâche utilise le module apt pour installer le paquet fail2ban. Le paramètre state : présent assure que fail2ban est installé sur le système cible. Si le paquet n'est pas déjà présent, ansible l'installera ; s'il l'est déjà, ansible n'entreprendra aucune action supplémentaire.

2.3 Préparation de l'environnement de déploiement

2.3.1 Infrastructure requise (machines, OS, réseau) :

Description des ressources matérielles et logicielles nécessaires

-Serveur de Déploiement : Un serveur dédié pour exécuter Ansible et gérer les déploiements automatiques. Il peut être une machine physique ou virtuelle avec des spécifications modestes (par exemple, 2CPU, 4GB de RAM).

-Cibles de Déploiement : Les serveurs ou machines virtuelles sur lesquels les systèmes de détection d'intrusions et autres logiciels seront déployés. Les spécifications varient selon les besoins du service

à exécuter (par exemple, 2-4CPU, 8-16GB de RAM pour des serveurs web ou de base de données).

Configuration réseau et exigences spécifiques
-Connectivité Réseau : Une connectivité réseau fiable entre le serveur de déploiement et les cibles. Cela inclut des configurations réseau stables avec des adresses IP fixes ou des noms de domaine configurés.
-Sécurité Réseau : Pare-feu et configurations de sécurité réseau pour protéger les communications entre les machines. Utilisation de SSH avec authentification par clé pour sécuriser les accès.

2.3.2 Configuration initiale des machines :

Installation du Système d'exploitation :
-Choix de la Distribution : Sélectionner une distribution linux adaptée aux besoins, telles qu'Ubuntu Server, Debian, CentOS, ou Red Hat Enterprise Linux (RHEL). Les distributions LTS (Long Term Support) sont recommandées pour la stabilité et les mises à jour de sécurité à long terme.
-Installation de Base :Télécharger l'image ISO de la distribution choisie depuis le site officiel.
-Graver l'image ISO sur un support d'installation (comme une clé USB) ou utiliser un gestionnaire de machines virtuelles.
-Démarrer l'installation en suivant les instructions de l'assistant d'installation.
-Configurer les paramètres de base (langue, fuseau horaire, partitionnement du disque).
-Mettre à jour les paquets pour s'assurer que le système est à jour avec les dernières versions et correctifs de sécurité avec la commande «sudo apt-get update && sudo apt-get upgrade»

2.3.3 Sécurité et permissions :

Configuration des permissions utilisateurs.
-Ajouter un Utilisateur Non-root : Pour des raisons de sécurité, il est recommandé d'effectuer les tâches administratives avec un utilisateur non-root doté de privilèges sudo.
-Configurer les Permissions sudo : Il est important de configurer les permissions sudo de manière appropriée pour éviter les risques de sécurité liés aux utilisateurs ayant des privilèges excessifs.

Sécurisation des communications (SSH, firewall).
-Gestion des Clés SSH : Pour se connecter en SSH, il est recommandé d'utiliser une paire de clés plutôt que d'utiliser un mot de passe.La communication SSH est établie sur la base de clés SSH ; cette pratique est conseillée, car elle permet un niveau d'authentification beaucoup plus sûr que l'authentification par mot de passe.
-Générer des Clés SSH : Créer une paire de clés SSH de type ecdsa pour l'utilisateur(dans notre cas le nodemanager) avec la commande «ssh-keygen -t ecdsa».

```
nodemanager@ansible:~$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/nodemanager/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nodemanager/.ssh/id_ecdsa
Your public key has been saved in /home/nodemanager/.ssh/id_ecdsa.pub
The key fingerprint is:
SHA256:3+dk146IjjAQlihkEeuomUY770z2Mhz2DFON1LqcWOk nodemanager@ansible
The key's randomart image is:
+---[ECDSA 256]---+
| =o   .          |
|o .. o .         |
|... = =          |
|o. . B .         |
|.o  B o S        |
|oo.* E   . .    .|
|++oo* o   . . + o|
|. *+.o o . . * + |
|  .+o.  ..o . o .|
+----[SHA256]-----+
nodemanager@ansible:~$
```

FIGURE 2.2 : Génération de clés ssh

Observation :
La commande «`ssh-keygen -t ecdsa`» permet de générer une paire de clés SSH utilisant l'algorithme ECDSA (Elliptic Curve Digital Signature Algorithm). Cette méthode d'authentification repose sur des courbes elliptiques, offrant une sécurité renforcée avec des clés plus courtes par rapport à RSA.

Les principaux avantages de cette approche sont :

- Sécuriser les connexions SSH en éliminant l'authentification par mot de passe.
- Générer une clé publique et une clé privée pour une authentification sécurisée.
- Améliorer la rapidité et la robustesse cryptographique par rapport aux clés RSA traditionnelles.

Ensuite nous avons ajoutez la clé publique de l'utilisateur sur les nodes avec la commande : «ansible -i inventaire.ini -m authorized_key -a 'user=user-ansible state=present key="{{lookup("file", "/home/user-
ansible/.ssh/id_ecdsa.pub") }}"' –user user-ansible –ask-pass –become –ask-become-pass all»



FIGURE 2.3 : Ajout de la clé publique sur les nodes

Observation :
La commande suivante permet d'ajouter une clé SSH publique à l'utilisateur `user-ansible` sur les hôtes définis dans le fichier d'inventaire :

```
<<ansible -i inventaire.ini -m authorized_key \ -a 'user=user-ansible
state=present key="{{ lookup("file", "/home/user-ansible/.ssh/id_ecdsa.
pub")}}"' \--user user-ansible --ask-pass --become --ask-become-pass all>>
```

Elle exécute les actions suivantes :

- Se connecte aux hôtes distants en tant que `user-ansible`.
- Lis la clé publique stockée localement dans «`/home/user-ansible/.ssh/id_ecdsa.pub`».
- Ajoute cette clé au fichier « `/.ssh/authorized_keys`» de `user-ansible` sur les hôtes cibles.
- Utilise «`-ask-pass`» pour demander le mot de passe SSH si l'authentification par clé n'est pas encore configurée.
- Utilise «`-become`» et «`-ask-become-pass`» pour exécuter la commande avec des privilèges sudo.

Cette approche permet d'automatiser la distribution des clés SSH pour une connexion sécurisée sans mot de passe.

Relancez la commande sans `-ask-pass` :

```
<<ansible -i inventaire.ini -m authorized_key -a 'user=user-ansible state=
present key="{{lookup("file", "/home/user-ansible/.ssh/id_ecdsa.pub") }}
"' --user user-ansible --become --ask-become-pass all>>
```

- `-m authorized_key` : Le module `authorized_key` permet d'ajouter ou de supprimer des clés SSH pour un utilisateur.

- `user=user-ansible` : L'utilisateur concerné est `user-ansible`.
- `state=present` : Indique d'ajouter la clé SSH à l'utilisateur.
- `key="lookup("file", "/home/user-ansible/.ssh/id_ecdsa.pub")"` : Utilise la fonction `lookup` pour lire le contenu de la clé publique SSH.

Un seul mot de passe vous a été demandé, celui pour `sudo`. Vous pouvez ainsi exécuter des commandes distantes avec Ansible sur les nœuds en utilisant les privilèges `sudo`.

2.4 Déploiement automatique de Fail2ban avec Ansible

2.4.1 Création et organisation des playbooks Ansible :

Pour structurer les playbooks et organiser les tâches de manière efficace, nous allons suivre une approche modulaire et bien organisée. Voici une structure recommandée pour un projet Ansible destiné à installer et configurer Fail2ban :

```
nodemanager@ansible:~$ cd Ansible/
nodemanager@ansible:~/Ansible$ tree fail2ban
fail2ban
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   ├── config.yml
│   ├── include-vars.yml
│   ├── installation.yml
│   └── main.yml
├── templates
│   ├── fail2ban.local.j2
│   ├── pam_jail.local.j2
│   └── ssh_jail.local.j2
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml

8 directories, 14 files
nodemanager@ansible:~/Ansible$ cd
```

FIGURE 2.4 : Arborescence du playbook

Observation :

L'arborescence du playbook Ansible montre comment les fichiers sont organisés pour automatiser le déploiement. Chaque dossier ou fichier a un rôle bien précis : les tâches, les variables et les configurations sont séparées pour rendre l'ensemble plus clair. Cette structure facilite la lecture, les modifications et l'ajout de nouvelles fonctionnalités si besoin. C'est aussi ce qui rend le projet plus facile à maintenir dans le temps.

Fichiers et contenu des playbooks

Fichier d'Inventaire (inventories/production/hosts) : Le fichier d'inventaire dans Ansible sert à définir les hôtes (machines) et les groupes d'hôtes sur lesquels les tâches automatisées seront exécutées. C'est un élément essentiel pour Ansible car il permet de spécifier les cibles de vos scripts et playbooks.

```
1 all:
2   host:
3     machinetest:
4       ansible_host: 192.168.19.131
5       ansible_user: root
```

FIGURE 2.5 : Fichier inventaire

Observation :

Ceci est un extrait d'un fichier d'inventaire pour Ansible, un outil d'automatisation

-host :Cela indique que ce qui suit sera une section contenant les détails des hôtes (machines) gérés par Ansible.

-machinetest : Ceci est un groupe d'hôtes. Dans ce cas, il y a un seul hôte nommé "machinetest". Vous pourriez avoir plusieurs hôtes sous différents groupes pour organiser vos machines de manière logique.

-ansible_host : 192.168.19.131 : Cette ligne spécifie l'adresse IP ou le nom d'hôte de la machine "machinetest". Ansible utilisera cette adresse pour se connecter à la machine.

-ansible_user : root :Cette ligne spécifie le nom d'utilisateur avec lequel Ansible se connectera à la machine. Dans ce cas, c'est l'utilisateur "root".

En résumé, ce fichier dit à Ansible de se connecter à la machine avec l'adresse IP 192.168.19.131 en utilisant l'utilisateur root et de la considérer comme faisant partie du groupe "machinetest".

Variables Globales

```yaml
---
# --- fail2ban ---
fail2ban_jail_dir: /etc/fail2ban/jail.d
fail2ban_filter_dir: /etc/fail2ban/filter.d
fail2ban_config_loglevel: 3 #1 (error), 2 (warn), 3 (info), or 4 (debug)
fail2ban_config_logtarget: SYSLOG #STDOUT STDERR SYSLOG FILE
fail2ban_config_destemail: lokossiigor@gmail.com
fail2ban_config_sender: fail2ban@example.com
fail2ban_config_sendername: fail2ban
fail2ban_config_ignoreip: #127.0.0.1/8 1.2.3.4
fail2ban_config_bantime: 3600
fail2ban_config_findtime: 1800
fail2ban_config_maxretry: 10


# --- Log Paths ---
#logs_gitea: /opt/gitea/data/logs/gitea.log
logs_sshd: /var/log/auth.log
logs_pam:  /var/log/auth.log

# --- Modules ---
fail2ban_modules:
  - ssh
  - pam
inventory: ../../inventaire.yaml

#---sftp
sftp_user: koffi
sftp_group: sftpgroup
sftp_user_password: koffisftp
sftp_chroot_dir: /sftp/{{ sftp_user }}
```

FIGURE 2.6 : Déclaration des variables globales

Observation :

-fail2ban_config_bantime : 3600

Description : Ce paramètre définit la durée pendant laquelle une adresse IP sera bannie (bloquée) après avoir dépassé le nombre maximal de tentatives de connexion échouées spécifié par maxretry.

Valeur : 3600 secondes (soit 1 heure).

Utilité : En réglant ce paramètre à 3600, nous configurons Fail2ban pour bannir une adresse IP pendant une heure après qu'elle a dépassé le nombre de tentatives autorisées. Cela aide à prévenir les attaques par force brute en rendant plus difficile pour les attaquants de continuer leurs tentatives.

-fail2ban_config_findtime : 1800

Description : Ce paramètre définit la fenêtre de temps pendant laquelle les tentatives de connexion échouées sont comptabilisées.

Valeur : 1800 secondes (soit 30 minutes).

Utilité : En réglant ce paramètre à 1800, nous configurons Fail2ban pour examiner les tentatives de connexion échouées sur une période de 30 minutes. Si une adresse IP dépasse le nombre de tentatives échouées autorisées (maxretry) dans ce laps de temps, elle sera bannie.

-fail2ban_config_maxretry : 10

Description : Ce paramètre définit le nombre maximal de tentatives de connexion échouées autori-

sées avant qu'une adresse IP ne soit bannie.
Valeur : 10 tentatives échouées.
Utilité : En réglant ce paramètre à 10, nous configurons Fail2ban pour bannir une adresse IP après 10 tentatives de connexion échouées dans la fenêtre de temps définie par findtime. Cela permet de tolérer quelques erreurs de frappe ou de mauvaise saisie de mots de passe par des utilisateurs légitimes, tout en protégeant contre des tentatives répétées de force brute.

2.4.1.1 Répertoire Tasks

-**Tâches principales** (Ansible/fail2ban/tasks/main.yml)

```
1  ---
2  - import_tasks: include-vars.yml
3
4  # fail2ban installation
5  - import_tasks: installation.yml
6
7  # fail2ban configuration
8  - import_tasks: config.yml
9
10 # sftp configuration
11 - import_tasks: sftp.yml
12
13
14 ...
```

FIGURE 2.7 : Importation des tâches

Observation :
Le playbook est structuré avec des tâches importées à partir d'autres fichiers YAML, ce qui permet de modulariser et d'organiser le code de manière plus lisible et maintenable.
Les tirets (—) indiquent le début d'un document YAML. Dans le contexte d'Ansible, cela signifie le début d'un playbook.
-import_tasks : 'include-vars.yml'
Description : Cette ligne importe et exécute les tâches définies dans le fichier 'include-vars.yml'.
Utilité : include-vars.yml est probablement utilisé pour définir des variables qui seront utilisées dans les autres tâches du playbook. Importer des variables en début de playbook permet de les rendre disponibles pour toutes les tâches suivantes.

#fail2ban installation
Description : Un commentaire indiquant que les tâches suivantes sont liées à l'installation de Fail2ban.
Utilité : Les commentaires aident à documenter le playbook et à le rendre plus compréhensible pour les autres utilisateurs (ou pour vous-même, lorsque vous le relisez plus tard).

-import_tasks : installation.yml
Description : Cette ligne importe et exécute les tâches définies dans le fichier 'installation.yml'.
Utilité : 'installation.yml' contient probablement les tâches nécessaires pour installer Fail2Ban sur les machines cibles, telles que l'installation des paquets requis.

#fail2ban configuration
Description : Un commentaire indiquant que les tâches suivantes sont liées à la configuration de Fail2ban.
Utilité : Ce commentaire sert de repère pour savoir que les tâches suivantes concernent la configuration de Fail2ban.

-import_tasks : config.yml
Description : Cette ligne importe et exécute les tâches définies dans le fichier config.yml.
Utilité : config.yml contient probablement les tâches nécessaires pour configurer Fail2Ban, comme la modification des fichiers de configuration avec des paramètres spécifiques (comme ceux que vous avez mentionnés précédemment).

#sftp configuration
Description : Un commentaire indiquant que les tâches suivantes sont liées à la configuration de SFTP (Secure File Transfer Protocol).
Utilité : Ce commentaire aide à identifier que les tâches suivantes concernent la configuration de SFTP.

-import_tasks : sftp.yml
Description : Cette ligne importe et exécute les tâches définies dans le fichier sftp.yml.
Utilité : sftp.yml contient probablement les tâches nécessaires pour configurer SFTP sur les machines cibles, comme la création de répertoires SFTP, la configuration des permissions, et la modification des fichiers de configuration SSH pour activer SFTP.

Tâches d'installation (Ansible/fail2ban/tasks/installation.yml)

```
1  ---
2    - name: Install fail2ban
3      apt:
4        name: fail2ban
5        state: present
6
7    #- name: Install fail2ban-systemd
8    #  apt:
9    #    name: fail2ban-systemd
10   #  state: present
11
12   - name: Enable and Start fail2ban
13     systemd:
14       name: fail2ban
15       state: restarted
16       enabled: yes
17 ...
```

FIGURE 2.8 : Playbook d'installation fail2ban

Observation :

-**Tâche 1** : Installer Fail2ban

-**name** : Install fail2ban : Cette ligne définit le nom de la tâche. Elle permet de comprendre facilement ce que fait la tâche.

-**apt** : Indique que la tâche utilise le module apt d'Ansible, qui est utilisé pour gérer les paquets sur les systèmes basés sur Debian (comme Ubuntu).

name : fail2ban : Spécifie le nom du paquet à installer, ici fail2ban. **state** : present : Indique que le paquet doit être présent sur le système. Si le paquet n'est pas déjà installé, Ansible l'installera.

-**Tâche commentée** : Installation de fail2ban-systemd (optionnelle)

Cette section est commentée, donc elle ne sera pas exécutée.

Commentaire : Les lignes commencent par #, indiquant qu'elles sont des commentaires et seront ignorées par Ansible.

-**Installation de fail2ban-systemd** : Si décommentée, cette tâche installerait le paquet fail2ban-systemd. Cela pourrait être nécessaire si une version spéciale de Fail2ban ou une intégration spécifique à systemd est requise.

-**Tâche 2** : Activer et démarrer Fail2ban

-**name** : Enable and Start fail2ban : Nom de la tâche, indiquant qu'elle activera et démarrera le service Fail2ban.
-**systemd** : Utilise le module systemd d'Ansible pour gérer les services via systemd.
name : fail2ban : Spécifie le nom du service, ici fail2ban.
state : restarted : Indique que le service doit être redémarré. Cela signifie que si le service est déjà en cours d'exécution, il sera redémarré pour appliquer toute nouvelle configuration ; s'il n'est pas en cours d'exécution, il sera démarré.
enabled : yes : Indique que le service doit être activé au démarrage du système. Cela signifie que Fail2Ban démarrera automatiquement lors des prochains démarrages du système.

2.4.2 Récapitulatif et contexte

-Installation :La première tâche installe Fail2Ban en s'assurant que le paquet fail2ban est présent sur le système.

-Service management : La deuxième tâche utilise systemd pour s'assurer que Fail2ban est activé et en cours d'exécution, redémarrant le service si nécessaire.

Ce type de configuration est couramment utilisé pour automatiser l'installation et la configuration de services sur des serveurs, garantissant que les logiciels nécessaires sont installés et correctement configurés pour démarrer automatiquement avec le système.

Tâches de configuration(Ansible/fail2ban/tasks/config.yml)

```
1  ---
2    - name: Copy default Jail
3      template:
4        src: templates/fail2ban.local.j2
5        dest: "{{ fail2ban_jail_dir }}/fail2ban.local"
6      ignore_errors: yes
7
8    #- name: Copy filters
9    #  template:
10   #    src: templates/{{ item }}_filter.conf.j2
11   #    dest: "{{ fail2ban_jail_dir }}/{{ item }}.conf"
12   # with_items: "{{ fail2ban_modules }}"
13   #  ignore_errors: yes
14
15   - name: Copy jails
16     template:
17       src: templates/{{ item }}_jail.local.j2
18       dest: "{{ fail2ban_jail_dir }}/{{ item }}.local"
19     with_items: "{{ fail2ban_modules }}"
20     ignore_errors: yes
21     notify:
22     - Restart Fail2Ban
23 ...
```

FIGURE 2.9 : Playbook de configuration fail2ban

Observation :

-**Tâche 1** : Copier le fichier Jail par défaut

- **name** : Copy default Jail : Nom de la tâche, indiquant qu'elle copie le fichier Jail par défaut.

-**template** : : Utilise le module template d'Ansible pour copier un fichier modèle et le traiter avec les variables définies.

-**src** : templates/fail2ban.local.j2 : Spécifie le chemin source du fichier modèle. Le fichier modèle est généralement un fichier Jinja2 (.j2) qui permet d'insérer des variables et des structures de contrôle.

-**dest** : " fail2ban_jail_dir /fail2ban.local" : Spécifie le chemin de destination où le fichier sera copié. La variable fail2ban_jail_dir est interpolée pour déterminer le répertoire cible.

ignore_errors : yes : Indique que la tâche doit continuer même si elle rencontre des erreurs. Cela peut être utile si des erreurs non critiques sont prévues.

-**Tâche Commentée** : Copier les filtres (optionnelle)

-**Tâche 2** : Copier les fichiers Jail

-**name** : Copy jails : Nom de la tâche, indiquant qu'elle copie les fichiers Jail spécifiques pour chaque module.

-**template** : Utilise le module template pour copier des fichiers modèles et les traiter avec les variables définies.

-**src** : templates/ item _jail.local.j2 : Spécifie le chemin source du fichier modèle pour chaque élément dans fail2ban_modules.
-**dest** : " fail2ban_jail_dir item .local" : Spécifie le chemin de destination où le fichier sera copié pour chaque élément. La variable item est interpolée pour déterminer le nom du fichier.
-**with_items** : " fail2ban_modules " : Utilise une boucle pour répéter la tâche pour chaque élément de la liste fail2ban_modules.
-**ignore_errors** : yes : Indique que la tâche doit continuer même si elle rencontre des erreurs.
-**notify** : Spécifie une action à notifier si la tâche est modifiée.
-**Restart Fail2ban** : Notifie Ansible pour redémarrer le service Fail2ban si cette tâche a apporté des modifications.

Explication des concepts
-**Module template :** Le module template est utilisé pour copier des fichiers modèles (Jinja2) tout en permettant l'injection de variables Ansible dans ces fichiers.
-**Variables et interpolation :** fail2ban_jail_dir et item sont des variables qui seront remplacées par leurs valeurs actuelles au moment de l'exécution. with_items : " fail2ban_modules " : Permet de passer une liste de valeurs à la tâche, ce qui permet de répéter l'opération pour chaque élément de la liste.
-**Gestion des erreurs :** ignore_errors : yes : Permet à la tâche de continuer même si une erreur est rencontrée, utile pour éviter que le playbook ne s'arrête sur des erreurs non critiques.
-**Handlers et notifications** :
notify : Spécifie qu'une action doit être exécutée si la tâche a apporté des modifications. Dans ce cas, il s'agit de redémarrer Fail2ban pour appliquer les nouvelles configurations.

Tâches include-vars.yml(Ansible/fail2ban/tasks/include-vars.yml)

```
1 ---
2 - name: check if local var file "/etc/ansible/{{ ansible_nodename }}.yml" exists
3   stat:
4     path: "/etc/ansible/{{ ansible_nodename }}.yml"
5   register: local_varfile
6 - name: Load a variable file "/etc/ansible/{{ ansible_nodename }}.yml"
7   include_vars: "/etc/ansible/{{ ansible_nodename }}.yml"
8   when: local_varfile.stat.exists
```

FIGURE 2.10 : Include vars

Observation :
Tâche 1 : Vérifier l'existence d'un fichier de variables local
-Nom de la tâche : check if local var file "/etc/ansible/ ansible_nodename .yml" exists
Cette ligne donne un nom à la tâche, indiquant clairement ce qu'elle fait : vérifier si un fichier de variables local existe.

-**Module stat** : Utilise le module stat d'Ansible pour obtenir des informations sur le fichier spécifié.

-**path** : "/etc/ansible/ ansible_nodename .yml" : Spécifie le chemin du fichier à vérifier. La variable ansible_nodename est une variable factuelle d'Ansible qui contient le nom du nœud (nom de

la machine). Le chemin complet du fichier sera donc basé sur le nom de la machine.

-**Register : local_varfile** : Enregistre les résultats de la tâche stat dans une variable nommée local_varfile. Cette variable contiendra des informations sur le fichier, y compris s'il existe ou non.

-**Tâche 2** : Charger un fichier de variables local

-**Nom de la tâche** : Load a variable file "/etc/ansible/ ansible_nodename .yml" Cette ligne donne un nom à la tâche, indiquant qu'elle charge un fichier de variables.

-**Module include_vars** : Utilise le module include_vars d'Ansible pour inclure les variables définies dans un fichier.

-**include_vars** : "/etc/ansible/ ansible_nodename .yml" : Spécifie le chemin du fichier de variables à charger. Encore une fois, ansible_nodename est utilisé pour dynamiser le nom du fichier en fonction du nom de la machine.

-**Condition when** : **when** : local_varfile.stat.exists : Cette condition indique que la tâche include_vars ne doit être exécutée que si le fichier existe. La condition utilise la variable local_varfile enregistrée par la tâche précédente et vérifie la clé stat.exists pour voir si le fichier existe.

Rôle des tâches

Vérification de l'existence du fichier : La première tâche utilise le module stat pour vérifier si le fichier de variables spécifique à la machine existe. Le résultat de cette vérification est enregistré dans la variable local_varfile. Chargement Conditionnel des Variables :

La deuxième tâche utilise le module 'include_vars' pour charger les variables à partir du fichier spécifié, mais seulement si le fichier existe. La condition when empêche la tâche de provoquer une erreur si le fichier n'est pas présent.

Utilité : Ces tâches permettent de charger dynamiquement des configurations spécifiques à chaque machine sans provoquer d'erreurs si certains fichiers de configuration sont manquants. Cela permet de personnaliser facilement les configurations pour différents nœuds dans votre infrastructure tout en maintenant un playbook générique et réutilisable.

2.4.2.1 Répertoire templates

Le répertoire templates dans un projet Ansible sert à stocker des fichiers de modèles (templates) qui sont utilisés pour générer des fichiers de configuration dynamiques sur les hôtes gérés. Les fichiers de modèles sont généralement écrits en utilisant la syntaxe Jinja2, qui permet l'insertion de variables et l'utilisation de structures de contrôle telles que des boucles et des conditions.

Fichier fail2ban.local.j2

```
[DEFAULT]
loglevel = {{ fail2ban_config_loglevel }}
logtarget = {{ fail2ban_config_logtarget }}
destemail = {{ fail2ban_config_destemail }}
sendername = {{ fail2ban_config_sendername }}
sender = {{ fail2ban_config_sender }}
ignoreip = {{ fail2ban_config_ignoreip }}
bantime = {{ fail2ban_config_bantime }}
findtime = {{ fail2ban_config_findtime }}
maxretry = {{ fail2ban_config_maxretry }}
```

FIGURE 2.11 : Fail2ban.local.j2

Observation :
Ce bloc de configuration est utilisé dans Ansible pour générer un fichier de configuration pour le service Fail2ban. Chaque ligne spécifie une variable entre doubles accolades (...), ce qui indique qu'elle sera remplacée par sa valeur respective au moment de l'exécution du playbook Ansible. Voici une explication détaillée de chaque ligne :

Explication ligne par ligne

C'est une section dans le fichier de configuration où les paramètres par défaut pour le service Fail2ban sont définis.

- **loglevel** : Niveau de journalisation pour Fail2ban.
- **fail2ban_config_loglevel** : Cette syntaxe Jinja2 indique qu'une variable fail2ban_config_loglevel sera insérée ici lors de la génération du fichier de configuration.
- **logtarget** : Cible de journalisation pour les messages de Fail2ban.
- **fail2ban_config_logtarget** : Variable définissant la destination où les journaux seront envoyés.
- **destemail** : Adresse email à laquelle les notifications de fail2Ban seront envoyées.
- **fail2ban_config_destemail** : Variable spécifiant l'adresse email de destination pour les notifications.
- **sendername** : Nom de l'expéditeur pour les emails envoyés par Fail2ban.
- **fail2ban_config_sendername** : Variable indiquant le nom de l'expéditeur des emails.
- **sender** : Adresse email de l'expéditeur pour les emails envoyés par Fail2Ban.
- **fail2ban_config_sender** : Variable définissant l'adresse email de l'expéditeur.

- **ignoreip** : Liste d'adresses IP à ignorer par Fail2Ban (par exemple, des adresses IP internes ou fiables).
- **fail2ban_config_ignoreip** : Variable contenant la liste des adresses IP à ignorer.
- **bantime** : Durée pendant laquelle une adresse IP est bannie par Fail2ban (en secondes).
- **fail2ban_config_bantime** : Variable spécifiant la durée du bannissement.
- **findtime** : Période de temps sur laquelle les tentatives de connexion sont analysées pour le bannissement (en secondes).
  fail2ban_config_findtime : Variable définissant cette période de temps.
- **maxretry** : Nombre maximum de tentatives de connexion autorisées avant le bannissement.
- **fail2ban_config_maxretry** : Variable spécifiant ce nombre maximum.

```
1 # SSHD Configuration
2
3 Port 22
4 Protocol 2
5
6 # SFTP configuration
7 Subsystem sftp internal-sftp
8
9 Match Group {{ sftp_group }}
10     ChrootDirectory {{ sftp_chroot_dir }}
11     ForceCommand internal-sftp
12     AllowTcpForwarding no
13     X11Forwarding no
14
```

FIGURE 2.12 : sshd.config.j2

Observation :
-# SSHD Configuration : C'est un commentaire indiquant que les lignes suivantes concernent la configuration du service SSHD (SSH Daemon).
-Port 22 : Cela définit le port sur lequel le serveur SSH écoute. Par défaut, SSH utilise le port 22 pour les connexions.
-Protocol 2 : Spécifie que le serveur SSH doit utiliser le protocole SSH version 2. Ce protocole est plus sécurisé que la version 1 et est largement utilisé.
-Commentaire de section SFTP
-# SFTP configuration : C'est un commentaire indiquant que les lignes suivantes concernent la configuration du service SFTP, qui utilise SSH pour le transfert sécurisé de fichiers.

-Subsystem sftp internal-sftp : Cela configure le sous-système SFTP pour utiliser le service SFTP interne (internalsftp). Cela signifie que le serveur SSH gère lui-même le service SFTP plutôt que de le déléguer à un autre processus.
-Match Group sftp_group : Cela commence une section de correspondance (Match) qui spécifie des configurations spécifiques qui s'appliquent uniquement aux utilisateurs membres du groupe spécifié par la variable sftp_group.
-ChrootDirectory sftp_chroot_dir : Configure le répertoire racine (ChrootDirectory) dans lequel les utilisateurs SFTP seront confinés. Cela limite les utilisateurs à ce répertoire spécifique pour leurs opérations SFTP.
-ForceCommand internal-sftp : Force l'exécution de la commande spécifiée (ForceCommand) pour tous les utilisateurs correspondant à la règle. Dans ce cas, cela force l'utilisation du service SFTP interne.
-AllowTcpForwarding no : Désactive (no) la possibilité de transférer des connexions TCP via SSH. Cela sécurise davantage le serveur en empêchant le transfert non autorisé de trafic réseau.
-X11Forwarding no : Désactive (no) le transfert d'affichage X11 via SSH. Cela est couramment désactivé pour des raisons de sécurité, à moins que l'utilisation de X11 soit explicitement nécessaire.

```
1 [sshd]
2 enabled = true
3 backend = systemd
4 port = ssh
5 maxretry = 4
6 findtime = 600
7 bantime = 3600
8 action = iptables-allports
9 filter = sshd
10 logpath  = {{ logs_sshd }}
```

FIGURE 2.13 : ssh_jail.local.j2

Observation :
Ce bloc de configuration est destiné à configurer les paramètres spécifiques pour le service SSHD (Secure Shell Daemon)
-Section [sshd] : Cela indique qu'il s'agit d'une section spécifique dans un fichier de configuration où les paramètres pour le service SSHD sont définis.
-enabled = true : Cela indique que le service SSHD est activé (true). Cela signifie que le service SSHD

sera démarré et écoutera les connexions entrantes sur le port spécifié.
-backend = systemd : Spécifie le backend à utiliser pour le service SSHD. Dans ce cas, systemd indique que le service SSHD est géré par systemd sur les systèmes Linux modernes.
-port = ssh : Définit le port sur lequel le service SSHD écoute. Ici, ssh est probablement une référence à l'utilisation du port standard SSH, qui est le port 22 par défaut.
-maxretry = 4 : Définit le nombre maximum de tentatives de connexion autorisées avant que Fail2Ban ne prenne des mesures, comme le bannissement de l'adresse IP.
-findtime = 600 : Spécifie la période de temps (en secondes) pendant laquelle les tentatives de connexion sont surveillées pour atteindre le nombre maximum d'essais (maxretry). Dans ce cas, il surveille les tentatives sur une période de 600 secondes (10 minutes).
-bantime = 3600 : Définit la durée (en secondes) pendant laquelle une adresse IP est bannie après avoir dépassé le nombre maximum d'essais (maxretry). Ici, l'adresse IP est bannie pendant 3600 secondes (1 heure).
-action = iptables-allports : Spécifie l'action à prendre lorsque les critères de bannissement sont rencontrés. Dans ce cas, iptables-allports indique que tous les ports doivent être bannis en utilisant iptables, un pare-feu Linux couramment utilisé.
-filter = sshd : Indique quel filtre Fail2ban doit utiliser pour surveiller les tentatives de connexion au service SSHD. Le filtre sshd est spécifié ici, ce qui signifie que Fail2Ban surveillera les journaux associés au service SSHD pour détecter les tentatives de connexion infructueuses.
-logpath = logs_sshd : Utilise une variable Jinja2 (logs_sshd) pour spécifier le chemin du fichier journal où les tentatives de connexion SSHD sont enregistrées. Cette variable serait remplacée par un chemin de journal spécifique au moment de l'exécution du playbook Ansible.

```
1 [pam-generic.conf]
2 enabled = true
3 port = all
4 maxretry = 10
5 findtime = 3600
6 bantime = 900
7 action = iptables-allports
8 filter = pam-generic
9 logpath  = {{ logs_pam }}
```

FIGURE 2.14 : pam_jail.local.j2

Observation :

En utilisant ce modèle, Ansible peut générer dynamiquement le fichier /etc/fail2ban/jail.d/pam-generic.conf avec les valeurs spécifiées, adaptant ainsi la configuration de Fail2Ban pour PAM à chaque hôte ou groupe d'hôtes en fonction des besoins spécifiques de configuration et des chemins de journal définis.
-Section [pam-generic.conf] : Cela indique qu'il s'agit d'une section spécifique dans un fichier de configuration où les paramètres pour le module PAM (Pluggable Authentication Modules) sont définis.
-enabled = true : Cela indique que ce filtre est activé (true). Fail2Ban utilisera ces paramètres pour surveiller et protéger contre les attaques utilisant le système PAM.
-port = all : Définit le port sur lequel le filtre s'applique. Dans ce cas, all indique que le filtre surveillera toutes les tentatives de connexion, quel que soit le port.
-maxretry = 10 : Définit le nombre maximum de tentatives de connexion autorisées avant que Fail2Ban ne prenne des mesures, telles que le bannissement de l'adresse IP.
-findtime = 3600 : Spécifie la période de temps (en secondes) pendant laquelle les tentatives de connexion sont surveillées pour atteindre le nombre maximum d'essais (maxretry). Dans ce cas, il surveille les tentatives sur une période de 3600 secondes (1 heure).
-bantime = 900 : Définit la durée (en secondes) pendant laquelle une adresse IP est bannie après avoir dépassé le nombre maximum d'essais (maxretry). Ici, l'adresse IP est bannie pendant 900 secondes (15 minutes)
-action = iptables-allports : Spécifie l'action à prendre lorsque les critères de bannissement sont rencontrés. Dans ce cas, iptables-allports indique que tous les ports doivent être bannis en utilisant iptables, un pare-feu Linux couramment utilisé.
-filter = pam-generic : Indique quel filtre Fail2ban doit utiliser pour surveiller les tentatives de connexion au module PAM. Le filtre pam-generic est spécifié ici, ce qui signifie que Fail2Ban surveillera les journaux associés au module PAM pour détecter les tentatives de connexion infructueuses.
-logpath = logs_pam : Utilise une variable Jinja2 (logs_pam) pour spécifier le chemin du fichier journal où les tentatives de connexion au module PAM sont enregistrées. Cette variable serait remplacée par un chemin de journal spécifique au moment de l'exécution du playbook Ansible.

2.4.3 Tests de déploiement sur une machine unique :

Pour effectuer des tests de déploiement sur une machine unique avec Ansible assurez-vous d'avoir un inventaire d'hôtes. Créez un fichier d'inventaire (ou tout autre nom) dans le répertoire de votre projet Ansible. Ajoutez l'adresse IP ou le nom d'hôte de votre machine cible à cet inventaire.

```
all:
  host:
    machinetest:
      ansible_host: 192.168.19.131
      ansible_user: root
```

FIGURE 2.15 : Fichier inventaire

Assurez-vous que ansible_host est l'adresse IP de votre machine cible et ansible_user est le nom d'utilisateur avec lequel Ansible se connectera à cette machine.
Exécution des playbooks sur une machine de test

```
---
- name: Deploy Fail2ban
  hosts: fail2ban
  roles:
    - role: fail2ban

```

FIGURE 2.16 : Déploiement du palybook

Observation :
Ce fragment de code représente un playbook Ansible qui déploie fail2ban sur des hôtes spécifiés par le groupe fail2ban. Voici une explication détaillée ligne par ligne :

-name : Deploy Fail2ban Définit le nom du playbook Ansible. Ce nom est utilisé pour identifier le playbook lorsqu'il est exécuté. Dans ce cas, le playbook est nommé "Deploy Fail2ban".

-hosts : fail2ban Spécifie le groupe d'hôtes sur lesquels ce playbook doit être exécuté. Dans Ansible, les groupes d'hôtes sont des ensembles d'hôtes sur lesquels les tâches peuvent être exécutées simultanément. fail2ban est le nom du groupe d'hôtes défini dans votre fichier d'inventaire Ansible (hosts.yml ou similaire). Ce groupe peut contenir une ou plusieurs machines où Fail2Ban doit être déployé.

-roles : Définit la liste des rôles à appliquer dans ce playbook. Les rôles sont des ensembles de tâches, de fichiers de configuration et de modèles organisés de manière à être réutilisables et modulaires dans Ansible.

- role : fail2ban Spécifie le rôle Ansible à appliquer. Fail2ban est le nom du rôle à appliquer. Un rôle Ansible est généralement structuré dans un répertoire contenant des sous-répertoires prédéfinis (tasks, handlers, templates, etc.) qui contiennent des fichiers nécessaires à l'automatisation de la configuration, de l'installation et de la gestion de Fail2Ban.

Ensuite avec la commande «ansible-playbook -u root **./fail2ban/fail2ban.yml» lancer l'installation du palybook**

```
[sudo] Mot de passe de nodemanager :
nodemanager@ansible:~$ cd Ansible/
nodemanager@ansible:~/Ansible$ ansible-playbook -u root ./fail2ban/fail2ban.yml

PLAY [Deploy Fail2ban] *********************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************
ok: [192.168.19.131]

TASK [fail2ban : check if local var file "/etc/ansible/fail2ban.yml" exists] ***************************************************************
ok: [192.168.19.131]

TASK [fail2ban : Load a variable file "/etc/ansible/fail2ban.yml"] *************************************************************************
skipping: [192.168.19.131]

TASK [fail2ban : Install fail2ban] *********************************************************************************************************
ok: [192.168.19.131]

TASK [fail2ban : Enable and Start fail2ban] ************************************************************************************************
changed: [192.168.19.131]

TASK [fail2ban : Copy default Jail] ********************************************************************************************************
ok: [192.168.19.131]

TASK [fail2ban : Copy jails] ***************************************************************************************************************
ok: [192.168.19.131] => (item=ssh)
ok: [192.168.19.131] => (item=pam)

PLAY RECAP *********************************************************************************************************************************
192.168.19.131             : ok=6    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

nodemanager@ansible:~/Ansible$
```

FIGURE 2.17 : Déploiement effectuée

Observation :
On peut observer sur l'image que le playbook a bien été exécuté sur notre machine test.

2.5 Extension du déploiement à plusieurs machines

L'extension du déploiement à plusieurs machines permet d'automatiser la configuration et la gestion d'un ensemble de systèmes de manière centralisée. Ansible utilise un **fichier d'inventaire** pour définir les hôtes cibles et organiser leur gestion. Cette approche garantit une administration cohérente, réduit les erreurs humaines et facilite l'évolutivité de l'infrastructure.

Dans cette section, nous mettrons en place un inventaire Ansible permettant le déploiement de **Fail2Ban** sur deux machines de test. Nous détaillerons les configurations nécessaires pour assurer une exécution fluide des playbooks.

2.5.1 Mise en place de l'inventaire Ansible pour plusieurs machines

Dans cette section, nous définissons un inventaire Ansible comprenant deux machines de test :

- **test2** : Adresse IP `192.168.65.144`
- **test3** : Adresse IP `192.168.65.140`

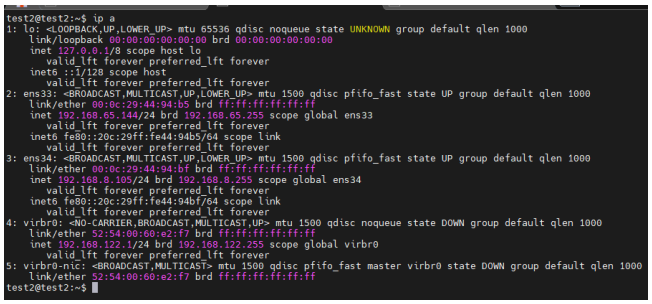


FIGURE 2.18 : Ip machine test2

Observation :
Avec la commande "ip a" nous constatons que la machine test2 a pour addres ip = 192.168.65.144

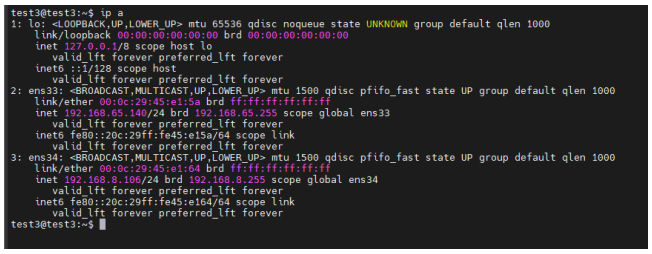


FIGURE 2.19 : Ip machine test3

Observation :
Avec la commande "ip a" nous constatons que la machine test3 a pour addres ip = 192.168.65.140

Avant de procéder au déploiement, nous devons vérifier la présence et l'état des services nécessaires sur les machines cibles :

- Vérification service **Fail2Ban** avec la commande : "systemctl status fail2ban"

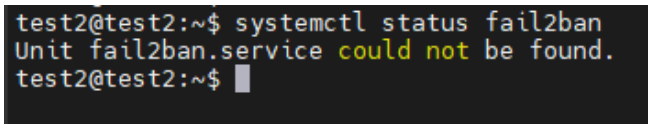


FIGURE 2.20 : Service fail2ban test2

Observation :
À l'aide de la commande suivante : systemctl status fail2ban nous avons constaté que **Fail2Ban** n'est pas installé sur la machine **test2**.

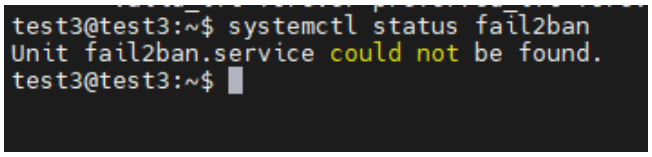


FIGURE 2.21 : Service fail2ban test3

Observation :
À l'aide de la commande suivante : "systemctl status fail2ban" nous avons constaté que **Fail2Ban** n'est pas installé sur la machine **test3**.



FIGURE 2.24 : Clé ssh

- Vérification de l'activation du service **SSH** avec la commande :

  "systemctl status ssh"

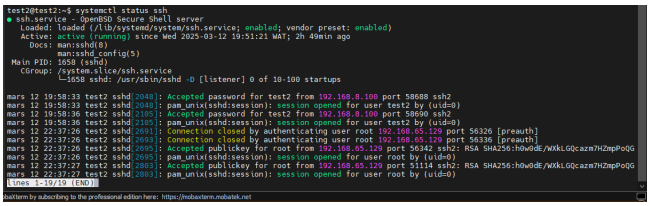


FIGURE 2.22 : Service ssh test2

Observation :
À l'aide de la commande : "systemctl status ssh" nous avons constaté que le service **ssh** est présent et activer sur la machine **test2**.

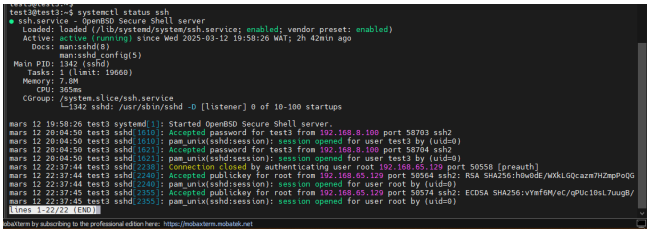


FIGURE 2.23 : Service ssh test3

Observation :
À l'aide de la commande : "systemctl status ssh" nous avons constaté que le service **ssh** est présent et activer sur la machine **test3**.

- Copie de la clé publique sur les nœuds

  Pour permettre une connexion sans mot de passe entre le gestionnaire Ansible et les machines cibles, nous devons copier la clé publique SSH. Tout d'abord, nous affichons la clé publique du node manager avec la commande : "cat ~/.ssh/id_rsa.pub"

  Ensuite, nous utilisons la commande suivante pour copier cette clé sur les machines cibles : ssh-copy-id ueser@machinecible

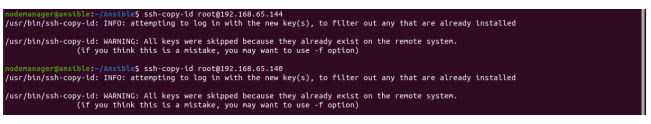


FIGURE 2.25 : Copie cle ssh

Observation :
ssh-copy-id root@192.168.65.144 pour copier la cle sur la machine test2
ssh-copy-id root@192.168.65.140 pour copier la cle sur la machine test3

- Déclaration des adresses IP dans le fichier d'inventaire et test de connectivité

  Avant de procéder au déploiement, nous devons déclarer nos machines cibles dans le fichier d'inventaire Ansible et vérifier leur accessibilité.

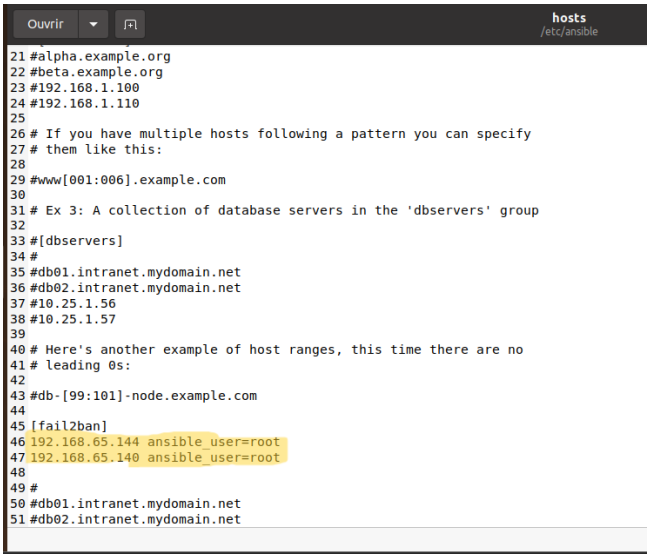


FIGURE 2.26 : Inventaire

Observation :
Les adresses IP sont bien déclarées dans le fichier d'inventaire.
Nous avons à la ligne 46 l'adresse IP `192.168.65.144` correspondant à la machine **test2**, et à la ligne 47 l'adresse IP `192.168.65.140` correspondant à la machine **test3**.

Une fois les clés SSH copiées et les adresses IP déclarées dans le fichier d'inventaire, nous testons la connexion entre les machines et le nœud Ansible en exécutant la commande suivante : ansible -m ping all



FIGURE 2.27 : Ping ssh

Observation :
Comme nous pouvons le voir sur la figure, les pings de connexion entre le nœud manager et les machines test2 et test3 passent avec succès. Cela confirme que les machines cibles sont accessibles et prêtes pour le déploiement.

2.5.2 Exécution des playbooks sur plusieurs machines :

Avec la commande «ansible-playbook -u root ./fail2ban/fail2ban.yml» lancer l'installation du palybook Cela installera le fail2ban ainsi que tous les services configurer dans notre playbook. Il est

important de vérifier si python est installé sur les machines teste pour que le déploiement soit effectif avec la commande : python3 –version

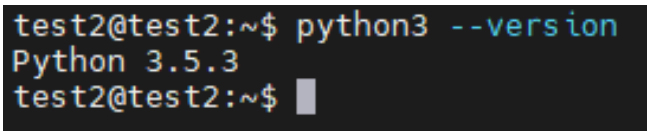


FIGURE 2.28 : Python test2

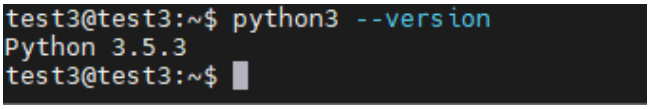


FIGURE 2.29 : Python test3

Nous pouvons lancer notre déploiement avec la commande : «ansible-playbook -u root ./fail2ban/fail2ban.yml»

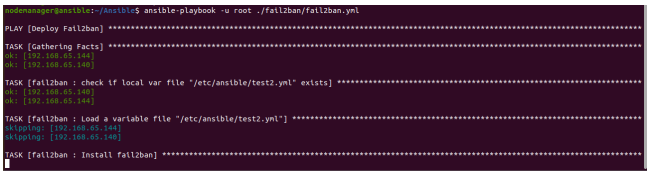


FIGURE 2.30 : Deploiement

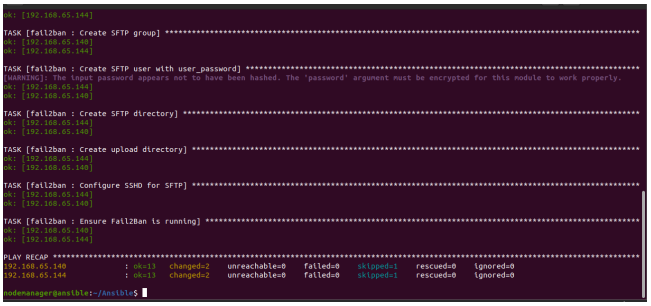


FIGURE 2.31 : Fin deploiement

Observation

Les images présentées montrent le succès du déploiement sur les machines de test. Après avoir configuré correctement les adresses IP dans l'inventaire Ansible et vérifié les prérequis nécessaires, tels que l'installation de Python et de SSH, la commande de déploiement a été exécutée avec succès. Les pings ont montré une connexion stable entre le node manager et les machines test2 et test3, confirmant ainsi que le déploiement a été effectué correctement.

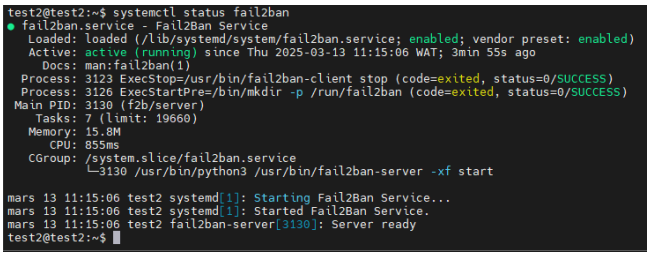


FIGURE 2.32 : Fail2ban machine test2

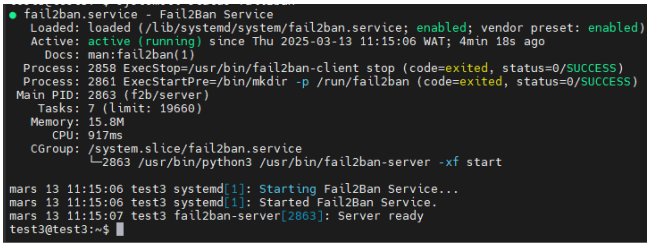


FIGURE 2.33 : Fail2ban machine test3

Les figures ci-dessus illustrent la confirmation que Fail2Ban est correctement installé sur les machines de test. Comme on peut le voir, les commandes 'systemctl status fail2ban' ont renvoyé des résultats indiquant que Fail2Ban est actif et fonctionne sur les deux machines, test2 et test3. Ces résultats confirment que l'installation a été effectuée avec succès sur les machines cibles.

Conclusion

Ce chapitre a posé les bases de l'automatisation du déploiement de Fail2Ban avec Ansible. Nous avons rappelé l'importance de détecter de manière proactive les tentatives d'intrusion telles que les attaques par force brute ou DDoS. Nous avons ensuite décrit comment Fail2Ban exploite les journaux système pour identifier ces menaces, et comment Ansible facilite leur déploiement à l'aide de playbooks. Ces éléments nous ont permis de construire une solution claire et reproductible pour renforcer la sécurité des machines ciblées.

Chapitre 3

Évaluation et optimisation du système déployé

Introduction

Ce chapitre s'est penché sur l'importance de l'évaluation continue du système Fail2Ban dans un environnement réel. Nous avons analysé la manière dont l'outil réagit face aux tentatives d'intrusion, notamment en observant ses journaux et en vérifiant l'efficacité des règles appliquées. Cette démarche vise à identifier les points d'amélioration possibles, à ajuster les paramètres selon les comportements observés, et à assurer une détection réactive des menaces. En affinant les configurations au fil des tests, nous cherchons à renforcer la capacité du système à répondre de façon cohérente et efficace aux attaques, tout en assurant sa stabilité et sa compatibilité avec une infrastructure en évolution. Cette phase de validation est indispensable pour garantir que le déploiement automatisé mis en place reste pertinent, fiable et adaptable face aux menaces actuelles comme futures.

3.1 Évaluation de la performance de Fail2ban

L'évaluation de Fail2ban sera menée selon une méthodologie structurée, intégrant des tests de simulation d'attaques et des analyses de logs réels. Cette approche permettra de mesurer objectivement la capacité de Fail2ban à détecter et à répondre aux tentatives d'intrusion, en tenant compte des différents paramètres de configuration.

3.1.1 Test d'Installation de Fail2ban

Ce test a pour but de vérifier que Fail2ban a bien été installé sur les machines cibles à l'aide d'Ansible. Il s'agit de s'assurer que le service est non seulement présent, mais également actif et prêt à être utilisé pour protéger le système contre les tentatives d'intrusion.

Après l'exécution du playbook, la commande «systemctl status fail2ban» a été utilisée sur les machines clientes. Les résultats ont confirmé que Fail2ban est installé et fonctionne correctement.

```
machinetest@fail2ban:~$ systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
     Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2024-06-28 23:04:45 CEST; 8min ago
       Docs: man:fail2ban(1)
   Main PID: 908 (f2b/server)
      Tasks: 7 (limit: 3395)
     Memory: 23.5M
     CGroup: /system.slice/fail2ban.service
             └─908 /usr/bin/python3 /usr/bin/fail2ban-server -xf start

juin 28 23:04:44 fail2ban systemd[1]: Starting Fail2Ban Service...
juin 28 23:04:45 fail2ban systemd[1]: Started Fail2Ban Service.
juin 28 23:04:57 fail2ban fail2ban-server[908]: Server ready
```

FIGURE 3.1 : Vérification de l'état de fail2ban

En s'assurant que Fail2ban est bien installé et que le service est actif, nous posons les bases nécessaires à sa mise en œuvre. Cette étape est indispensable avant de procéder à la configuration du système, car elle garantit que Fail2ban est prêt à détecter et bloquer les comportements suspects. Une installation réussie constitue donc un prérequis essentiel pour assurer une protection efficace contre les tentatives d'intrusion.

3.1.2 Test de configuration de Fail2ban

Nous réalisons ce test afin de vérifier que Fail2Ban est correctement configuré après son installation. Nous vérifions ainsi les fichiers de configuration et les paramètres définis pour garantir une protection efficace contre les tentatives d'intrusion.Pour cela, nous consultons les journaux de Fail2Ban en temps réel grâce à la commande : :«tail -f /var/log/fail2ban.log»Cela nous permet de détecter rapidement d'éventuelles erreurs ou anomalies.

```
machinetest@fail2ban:~$ tail -f /var/log/fail2ban.log
2024-06-28 23:04:56,527 fail2ban.jail           [908]: INFO    Creating new jail 'pam-generic.conf'
2024-06-28 23:04:57,355 fail2ban.jail           [908]: INFO    Jail 'pam-generic.conf' uses pyinotify {}
2024-06-28 23:04:57,365 fail2ban.jail           [908]: INFO    Initiated 'pyinotify' backend
2024-06-28 23:04:57,369 fail2ban.filter         [908]: INFO      maxRetry: 10
2024-06-28 23:04:57,370 fail2ban.filter         [908]: INFO      findtime: 3600
2024-06-28 23:04:57,370 fail2ban.actions        [908]: INFO      banTime: 900
2024-06-28 23:04:57,370 fail2ban.filter         [908]: INFO      encoding: UTF-8
2024-06-28 23:04:57,371 fail2ban.filter         [908]: INFO    Added logfile: '/var/log/auth.log' (pos = 0, hash = ce07bd36499360a2a4a963eca8c
992667e3f73ca)
2024-06-28 23:04:57,416 fail2ban.jail           [908]: INFO    Jail 'sshd' started
2024-06-28 23:04:57,422 fail2ban.jail           [908]: INFO    Jail 'pam-generic.conf' started
```

FIGURE 3.2 : Vérification paramètre de configuration sur le node

Les fichiers de configuration sont présents sur la machine cible avec les paramètres spécifiés.

3.1.3 Test de détection des intrusions

L'objectif de ce test est de vérifier la capacité de Fail2Ban à détecter les tentatives d'intrusion. Pour cela, nous analysons les journaux système afin d'identifier les activités suspectes et de déclencher les actions nécessaires pour les signaler.Pour simuler ces tentatives, nous avons utilisé le logiciel libre PuTTY sous Windows pour effectuer des connexions SSH à distance

```
machinetest@fail2ban:~$ sudo fail2ban-client status sshd
[sudo] Mot de passe de machinetest :
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `- Banned IP list:
```

FIGURE 3.3 : État de bannissement fail2ban par défaut

Par défaut voici comment se présente les journaux système.

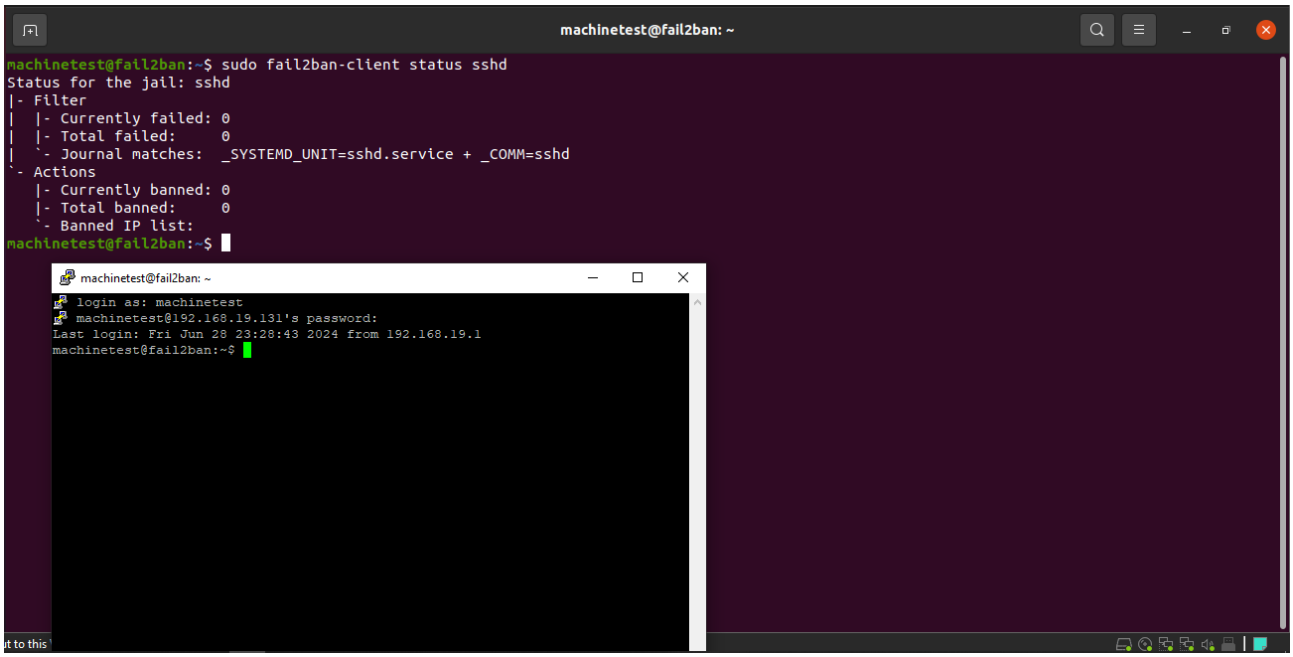


FIGURE 3.4 : Tentative de connexion

Voici comment elles se présentent après une tentative réussie : on peut remarquer qu'il n'y a aucune modification au niveau des prisons. Cette méthode nous a permis de générer des tentatives de connexion vers notre serveur cible, et l'on constate que la connexion a bien été établie.

3.1.4 Test de bannissement des adresses IP

L'objectif de ce test est de vérifier que Fail2Ban bannit correctement les adresses IP après plusieurs tentatives de connexion échouées. Nous validons ainsi la détection des intrusions, l'enregistrement des adresses suspectes, et l'application effective des règles de blocage pour empêcher de nouvelles tentatives depuis ces IP. Pour simuler ces connexions non autorisées, nous avons utilisé PuTTY sous Windows afin d'établir des sessions SSH à distance.
Voici les paramètres de bannissement de Fail2ban

```
[sshd]
enabled = true
backend = systemd
port = ssh
maxretry = 4
findtime = 600
bantime = 3600
action = iptables-allports
filter = sshd
logpath  = {{ logs_sshd }}
```

FIGURE 3.5 : Règle de bannissement définit pour ssh

Après quatre tentatives de connexion échouées, l'adresse IP est automatiquement bannie.
Les journaux par défaut de Fail2Ban nous permettent de vérifier ces tentatives.
Dans notre cas, après quatre échecs successifs depuis la machine <192.168.19.1>, nous constatons que cette adresse IP a bien été bloquée.

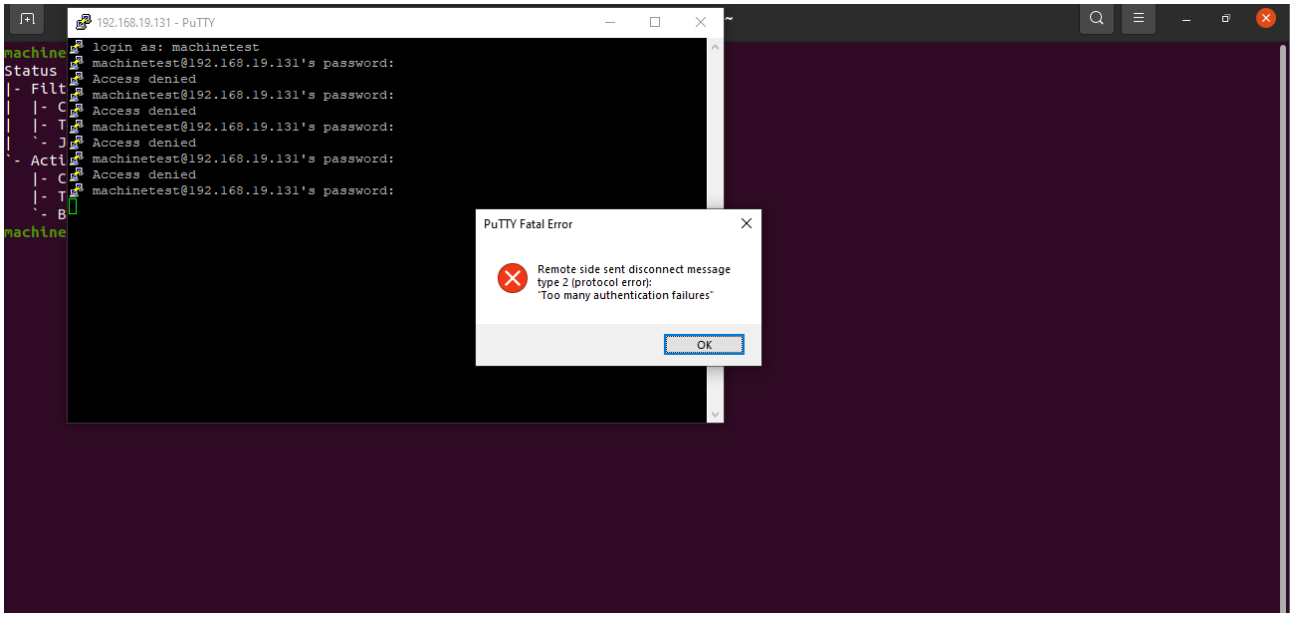


FIGURE 3.6 : Tentative de connexion après bannissement

Toute autre tentative de connexion provenant de cette adresse IP est alors rejetée pendant une durée minimale de 3600 secondes, soit 1 heure.

```
machinetest@fail2ban:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `- Banned IP list:
machinetest@fail2ban:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     6
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   192.168.19.1
machinetest@fail2ban:~$
```

FIGURE 3.7 : Vérification de l'ip bannit

3.1.5 Test de déploiement multi-machines

L'objectif de ce test est de vérifier la capacité d'Ansible à déployer Fail2Ban simultanément sur plusieurs machines. Pour cela, nous coordonnons les playbooks Ansible afin de garantir que Fail2Ban soit installé et configuré de manière cohérente sur chaque machine cible du réseau.
La création d'un fichier d'inventaire dans Ansible joue un rôle essentiel dans ce déploiement automatisé. Ce fichier centralise la gestion des machines en référençant leurs adresses IP, leurs configurations spécifiques et autres paramètres nécessaires.
Cette organisation assure une gestion efficace et sécurisée, en limitant l'application des configurations aux seules machines autorisées. Elle renforce ainsi la cohérence et la sécurité du déploiement.
Grâce à cette approche, le déploiement de Fail2Ban devient flexible et s'adapte aux besoins évolutifs des environnements informatiques complexes, tout en garantissant une protection uniforme contre les intrusions sur l'ensemble du réseau.

3.2 Analyse des résultats

L'analyse approfondie des résultats issus des tests de Fail2Ban déployé via Ansible révèle plusieurs points essentiels concernant son efficacité et sa fiabilité face aux intrusions.
Tout d'abord, nous constatons que Fail2Ban est particulièrement réactif et précis dans la détection des tentatives d'intrusion, grâce à une analyse rigoureuse des journaux système qui permet d'identifier rapidement les comportements suspects.
Ensuite, son mécanisme de bannissement temporaire des adresses IP s'est montré efficace pour contrer

les attaques en cours, tout en préservant la stabilité du système, même sous des charges simulées élevées.
Enfin, la gestion des faux positifs a été bien maîtrisée, garantissant que seules les adresses IP réellement problématiques sont bannies, ce qui permet de maintenir un accès légitime aux services.
Ces résultats confirment que Fail2Ban constitue une solution robuste et indispensable pour renforcer la sécurité des environnements informatiques contre les menaces persistantes.

3.3 Améliorations et perspectives futures

Pour améliorer et envisager l'avenir de Fail2ban dans mon mémoire, plusieurs axes peuvent être explorés.

3.3.1 Améliorations proposées

Pour enrichir les fonctionnalités de Fail2ban et explorer de nouvelles perspectives d'amélioration, voici plusieurs suggestions :

- **Intégration de l'apprentissage automatique** : Explorez l'ajout de techniques d'apprentissage automatique pour améliorer la détection des comportements malveillants et réduire les faux positifs. Cela pourrait inclure l'utilisation de modèles basés sur le comportement pour détecter les anomalies d'accès.
- **Support multi-plateforme renforcé** : Développez une compatibilité étendue avec une variété de plateformes et de systèmes d'exploitation, y compris des environnements cloud et conteneurisés, pour assurer une protection uniforme sur diverses infrastructures.
- **Amélioration de l'interface utilisateur** : Revoyez et améliorez l'interface utilisateur de Fail2ban pour la rendre plus conviviale et intuitive, facilitant ainsi la configuration, la surveillance et la gestion des politiques de sécurité.
- **Extension des actions de réponse automatisée** : Introduisez des actions de réponse automatisée plus avancées, telles que l'intégration avec des systèmes de gestion des incidents (SIEM), pour une réponse plus rapide et plus coordonnée aux incidents de sécurité.
- **Analyse Complète des logs et rapports** : Renforcez les capacités d'analyse des logs pour une surveillance continue et une génération de rapports détaillés sur les activités de bannissement, facilitant ainsi l'audit et la conformité réglementaire.
- **Détection préventive et adaptative** : Développez des capacités de détection préventive en identifiant les tendances d'attaques émergentes et en adaptant dynamiquement les règles de sécurité pour anticiper les menaces futures.
- **Intégration avec des outils de sécurité complémentaires** : Collaborez avec d'autres outils de sécurité complémentaires, tels que les pares-feux et les systèmes de détection d'intrusion (IDS), pour renforcer les défenses globales contre les attaques.

Ces améliorations visent à positionner Fail2ban comme une solution de sécurité avancée, capable de répondre aux défis croissants de sécurité informatique tout en offrant une gestion efficace et proactive des menaces potentielles.

3.3.2 Perspectives futures

Pour envisager les perspectives futures de Fail2ban, plusieurs axes de développement peuvent être explorés :

- **Intégration de l'intelligence artificielle et de l'apprentissage automatique** : Explorer l'intégration de techniques d'IA et d'apprentissage automatique pour améliorer la détection des anomalies et des comportements malveillants, permettant ainsi une réponse plus proactive et précise aux menaces.
- **Développement de fonctionnalités avancées de gestion des politiques de sécurité** : Élargir les capacités de gestion des politiques de sécurité pour permettre une configuration plus fine et granulaire des règles de bannissement, tout en simplifiant leur administration à travers une interface utilisateur améliorée.
- **Support étendu pour les environnements cloud et conteneurisés** : Adapter Fail2ban pour une intégration transparente avec les plateformes cloud et les architectures conteneurisées, en tenant compte des spécificités et des exigences de ces environnements dynamiques.
- **Automatisation renforcée avec des outils d'orchestration comme Ansible** : Continuer à développer des modules et des playbooks Ansible pour automatiser entièrement le déploiement, la configuration et la gestion de Fail2ban sur des infrastructures complexes et distribuées.
- **Amélioration de la gestion des incidents de sécurité** : Intégrer des capacités de gestion des incidents de sécurité pour faciliter la coordination et la réponse rapide aux incidents, en synchronisation avec d'autres outils de sécurité et de gestion des événements.
- **Renforcement de la collaboration et de l'interopérabilité** : Favoriser l'interopérabilité avec d'autres solutions de sécurité, telles que les pares-feux et les systèmes de détection d'intrusion (IDS), pour renforcer les défenses globales contre les attaques complexes et coordonnées.
- **Développement d'une communauté active et engagée** : Encourager le développement open source et maintenir une communauté active d'utilisateurs et de contributeurs pour favoriser l'innovation continue et la résolution collaborative des problèmes de sécurité.

En concentrant les efforts sur ces domaines, Fail2ban peut évoluer pour répondre aux besoins croissants de sécurité informatique, offrant ainsi une protection robuste et adaptative contre les menaces numériques émergentes.

Conclusion

En conclusion, ce chapitre a montré comment Fail2Ban, déployé automatiquement avec Ansible, analyse efficacement les journaux système pour détecter les comportements suspects et appliquer des mesures temporaires de protection. Les tests confirment sa robustesse et sa capacité à protéger le système sans compromettre sa stabilité. À l'avenir, l'intégration de nouvelles technologies et l'amélioration de l'interface utilisateur permettront d'accroître encore son efficacité et son adaptabilité face aux menaces informatiques.

Conclusion générale

En conclusion, ce mémoire présente notre travail sur le déploiement automatique de Fail2ban avec Ansible dans un contexte de sécurité informatique. Nous avons d'abord exploré les technologies de détection d'intrusions et souligné l'importance de solutions comme Fail2ban face aux menaces actuelles. Ensuite, nous avons détaillé son fonctionnement et mené des tests pour vérifier son efficacité et sa stabilité. Grâce à Ansible, le déploiement sur plusieurs machines s'est avéré simple et cohérent, facilitant l'automatisation des processus de sécurité. Les résultats confirment que Fail2ban protège efficacement sans nuire à la stabilité des systèmes. Pour l'avenir, nous envisageons d'intégrer des technologies émergentes comme l'intelligence artificielle et d'étendre la solution aux environnements cloud pour renforcer encore la protection et la gestion des menaces.

Webographie

[1] Avantages de fail2ban. https://www.provya.net/?d=2019/04/04/10/24/18-fail2ban-est-ce-vraiment-utile-partage-dexperience#:~:text=fail2ban%20prot%C3%A8ge%20efficacement%20contre%20ce,tentative%20de%20d%C3%A9ni%20de%20service., Consulté en avril 2024.

[2] Bénéfices attendus et défis. https://www.redhat.com/fr/blog/adventures-ansible-challenges-solutions-and%E2%80%A6-subversion, Consulté en avril 2024.

[3] Comparaison avec d'autres outils d'automatisation. https://www.redhat.com/fr/topics/automation/understanding-ansible-vs-terraform-puppet-chef-and-salt, Consulté en avril 2024.

[4] Définition et importance de la sécurité informatique. https://digital-solutions.konicaminolta.fr/infrastructure-informatique/article-securite-informatique-definition/#:~:text=La%20s%C3%A9curit%C3%A9%20informatique%20repr%C3%A9sente%20les,incontournable%20au%20sein%20des%20entreprises., Consulté en avril 2024.

[5] Fonctionnalités principales d'ansible. https://openclassrooms.com/fr/courses/2035796-utilisez-ansible-pour-automatiser-vos-taches-de-configuration, Consulté en avril 2024.

[6] Historique et évolution des ids. https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_d%C3%A9tection_d%27intrusion, Consulté en avril 2024.

[7] Importance de la détection d'intrusions. https://www.fortinet.com/fr/resources/cyberglossary/intrusion-detection-system#:~:text=Un%20syst%C3%A8me%20de%20d%C3%A9tection%20des,et%20des%20menaces%20de%20s%C3%A9curit%C3%A9., Consulté en juin 2024.

Annexe

Configuration de l'inventaire Ansible

L'inventaire Ansible liste les machines cibles sur lesquelles les tâches seront exécutées.

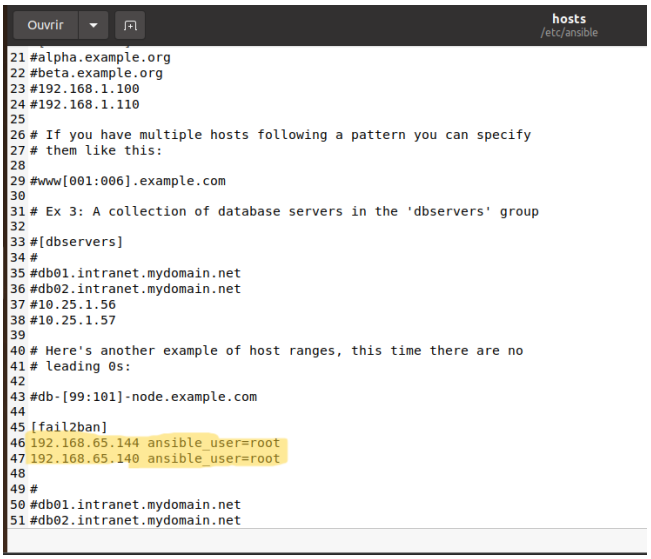


FIGURE 3.8 : Fichier d'inventaire Ansible

Observation :

- L'adresse IP `192.168.65.144` correspond à la machine **test2**.
- L'adresse IP `192.168.65.140` correspond à la machine **test3**.

Déploiement de Fail2ban avec Ansible

Le fichier YAML suivant a été utilisé pour installer Fail2ban automatiquement sur les serveurs cibles.

```yaml
---
  - name: Install fail2ban
    apt:
      name: fail2ban
      state: present

  #- name: Install fail2ban-systemd
  #  apt:
  #    name: fail2ban-systemd
  #  state: present

  - name: Enable and Start fail2ban
    systemd:
      name: fail2ban
      state: restarted
      enabled: yes
...
```

FIGURE 3.9 : Playbook d'installation de Fail2ban

Fichier fail2ban.local.j2

```
[DEFAULT]
loglevel = {{ fail2ban_config_loglevel }}
logtarget = {{ fail2ban_config_logtarget }}
destemail = {{ fail2ban_config_destemail }}
sendername = {{ fail2ban_config_sendername }}
sender = {{ fail2ban_config_sender }}
ignoreip = {{ fail2ban_config_ignoreip }}
bantime = {{ fail2ban_config_bantime }}
findtime = {{ fail2ban_config_findtime }}
maxretry = {{ fail2ban_config_maxretry }}
```

FIGURE 3.10 : Fail2ban.local.j2

Observation :

Ce bloc de configuration est utilisé dans Ansible pour générer un fichier de configuration pour le service Fail2ban. Chaque ligne spécifie une variable entre doubles accolades (...), ce qui indique qu'elle sera remplacée par sa valeur respective au moment de l'exécution du playbook Ansible.

- Activation de la protection SSH(règle de bannissement) :

```
1  [sshd]
2  enabled = true
3  backend = systemd
4  port = ssh
5  maxretry = 4
6  findtime = 600
7  bantime = 3600
8  action = iptables-allports
9  filter = sshd
10 logpath  = {{ logs_sshd }}
```

FIGURE 3.11 : ssh_jail.local.j2

Ce bloc de configuration est destiné à configurer les paramètres spécifiques pour le service SSHD

Vérification et tests

Une fois Fail2ban installé, nous avons vérifié son bon fonctionnement avec la commande suivante : systemctl status fail2ban

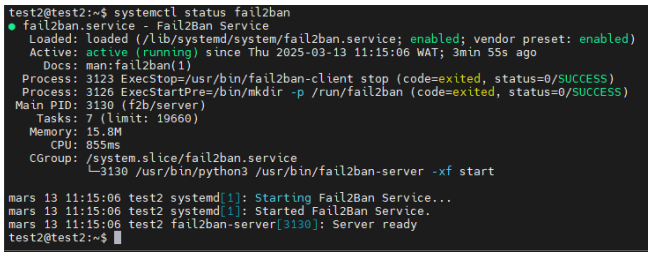


FIGURE 3.12 : Fail2ban machine test2

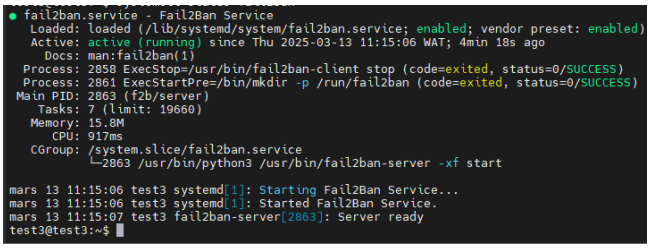


FIGURE 3.13 : Fail2ban machine test3

Observation : - Le service Fail2ban est bien actif et fonctionne correctement.

Simulation d'attaques et réponse de Fail2ban

Nous avons simulé plusieurs tentatives de connexion SSH infructueuses pour observer la réaction de Fail2ban.

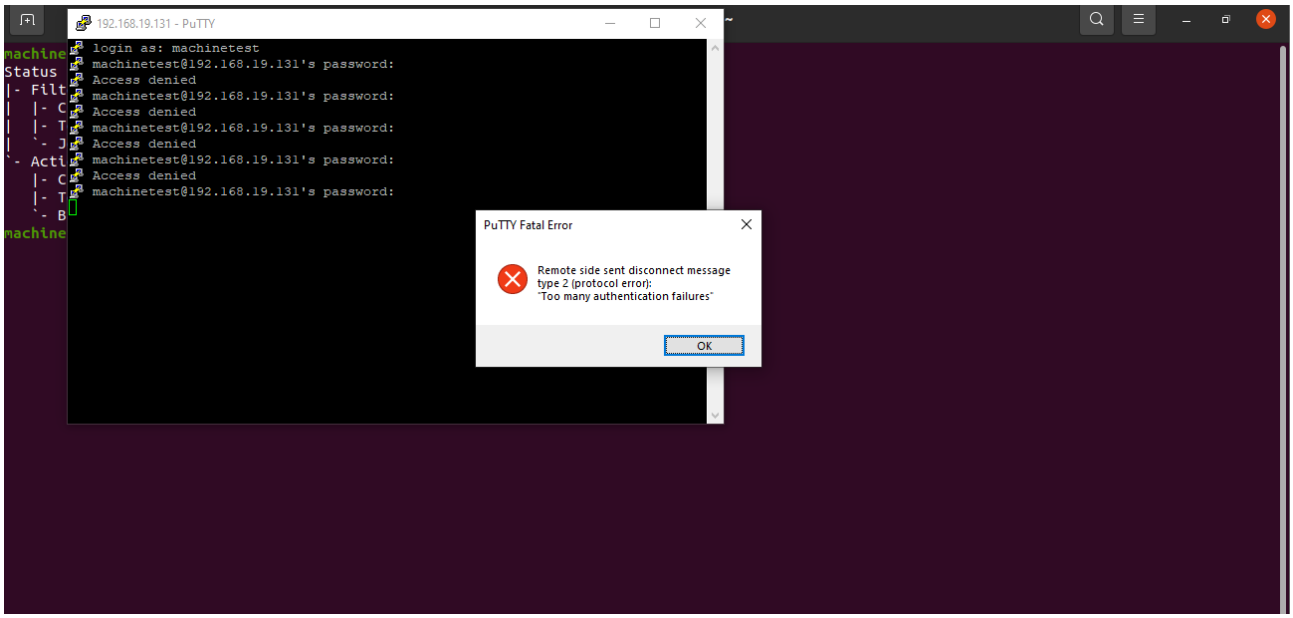


FIGURE 3.14 : Tentative de connexion après bannissement



FIGURE 3.15 : Vérification de l'ip bannit

Observation :

Après quatre tentatives de connexion échouées sur la même machine, l'adresse IP <192.168.19.1> a été correctement bannie.

Fail2ban détecte correctement les tentatives et applique le bannissement selon les règles définies.

Table des matières

Dédicace ii

Remerciements iii

Résumé iv

. iv

Abstract v

. v

Liste des figures vi

Liste des acronymes 1

Introduction 2

1 Revue de littérature 4

Introduction 4

1.1 Historique et évolution des IDS 4

1.1.1 Introduction aux Systèmes de Détection d'Intrusion (IDS) 4

1.1.2 Historique et développement des IDS 4

1.2 Typologies des IDS 5

1.2.1 Comparaison avec d'autres outils de détection d'intrusions 5

1.3 Contexte et choix de la solution 7

1.4 Fail2ban : Présentation et fonctionnalités 7

1.4.1 Introduction à Fail2ban 7

1.4.2 Fonctionnalités principales de Fail2ban 7

1.4.3 Avantages et limites de Fail2ban en tant qu'IDS 8

1.5 Ansible : Présentation et fonctionnalités 9

1.5.1 Introduction à Ansible 9

1.5.2 Fonctionnalités principales d'Ansible 9

1.5.3 Comparaison avec d'autres outils d'automatisation 10

1.6 Déploiement automatique et sécurité informatique 11

1.6.1 Importance du déploiement automatique dans la sécurité informatique 11

1.7 Bénéfices attendus et défis 12

1.7.1 **Bénéfices attendus** 12

1.7.2 **Défis** 12

Conclusion 12

2 Notions, outils utilisés et déploiement **14**
Introduction . . . 14
2.1 Notions de base . . . 14
2.1.1 Définition et importance de la sécurité informatique . . . 14
2.1.2 Principaux objectifs de la sécurité informatique . . . 14
2.1.3 Autres aspects de la sécurité informatique . . . 15
2.1.3.1 Types de menaces et attaques courantes . . . 15
2.1.4 Importance de la détection d'intrusions . . . 16
2.1.4.1 Rôle de la détection d'intrusions dans la protection des systèmes . . . 16
2.1.4.2 Avantages de la détection proactive des intrusions . . . 16
2.1.5 Concepts de base de Fail2Ban et Ansible . . . 16
2.1.5.1 Fail2Ban . . . 16
2.1.5.2 Ansible . . . 16
2.2 Présentation des outils utilisés . . . 17
2.2.1 Ansible : . . . 17
2.2.1.1 Installation et configuration de base . . . 17
2.2.1.2 Concepts de playbooks, rôles et inventaires : . . . 17
2.2.2 Fail2ban : . . . 18
2.2.2.1 Installation et configuration de base . . . 18
2.3 Préparation de l'environnement de déploiement . . . 18
2.3.1 Infrastructure requise (machines, OS, réseau) : . . . 18
2.3.2 Configuration initiale des machines : . . . 19
2.3.3 Sécurité et permissions : . . . 19
2.4 Déploiement automatique de Fail2ban avec Ansible . . . 22
2.4.1 Création et organisation des playbooks Ansible : . . . 22
2.4.1.1 Répertoire Tasks . . . 26
2.4.2 Récapitulatif et contexte . . . 29
2.4.2.1 Répertoire templates . . . 32
2.4.3 Tests de déploiement sur une machine unique : . . . 37
2.5 Extension du déploiement à plusieurs machines . . . 39
2.5.1 Mise en place de l'inventaire Ansible pour plusieurs machines . . . 39
2.5.2 Exécution des playbooks sur plusieurs machines : . . . 42

3 Évaluation et optimisation du système déployé **45**
Introduction . . . 45
3.1 Évaluation de la performance de Fail2ban . . . 45
3.1.1 Test d'Installation de Fail2ban . . . 45
3.1.2 Test de configuration de Fail2ban . . . 46
3.1.3 Test de détection des intrusions . . . 46
3.1.4 Test de bannissement des adresses IP . . . 47
3.1.5 Test de déploiement multi-machines . . . 49
3.2 Analyse des résultats . . . 49
3.3 Améliorations et perspectives futures . . . 50
3.3.1 Améliorations proposées . . . 50
3.3.2 Perspectives futures . . . 51

Conclusion 51

Conclusion **52**

Webographie **53**

Annexe **54**

Table des matières **59**