3 Configurer SSH avec Ansible

Fichiers sources :
- **Dossier "ansible/3"**

Nous allons apprendre à :
- Utiliser Ansible pour durcir les accès SSH à notre VM.
  - Désactiver l'authentification SSH par mot de passe
  - Demander une authentification par clé publique
  - Activer une authentification multi-facteurs (2FA) par SSH pour l'utilisateur florian

Premiers pas avec l' authentification par clé publique
- Rappels sur les infrastructures à clés publiques/privées
  - Concept crypto asymétrique
  - Création paire de clés
  - Challenge SSH
  - https://www.ssh.com/academy/ssh/
- Génération d'une paire de clés asymétriques avec une passphrase pour la clé privée
  - ***ssh-keygen -t rsa -f ~/.ssh/devsecops -C devsecops***
    - On créé une paire de clés RSA nommées devsecops
      - Si aucun nom spécifié, la valeur par défaut est id_rsa
      - Attention ! Cela remplacera votre clé par défaut (d'où l'intérêt de spécifier un nom).
    - Le flag -C ajoute un commentaire à la fin de la clé pour aider à l'identifier par un humain.
    - Lorsque demandé, saisissez une passphrase forte pour protéger votre clé privée;
    - Conservez cette passphrase à l'abri, car sans elle votre clé privée devient inutilisable !
    - Les clés sont stockées sous forme de fichiers dans le dossier ~/.ssh/
- Envoyer la clé publique vers la VM grâce à Ansible
  - Présentation fichier authorized-keys
    - Contient liste clés publiques utilisables par le serveur SSH pour authentifier cet utilisateur.
  - Ouvrir le dossier "ansible/3/"
  - Nous allons copier la clé publique locale créée au chapitre précédent et l'ajouter à la fin du fichier /home/florian/.ssh/authorized_keys sur la VM.
    - Ouvrir le fichier ansible/3/authorized_keys.yml
    - state: present (on veut ajouter et non supprimer le fichier)
    - La fonction Ansible "lookup" récupère des données à partir de sources externes en fonction du plugin défini en premier argument.
      Ici "lookup" utilise le plugin "file" pour lire le contenu de devssecops.pub
    - Le chemin d'accès à ce fichier est construit avec le plugin "env" (qui récupère des variables d'environnement) et le signe "+" qui permet de concaténer.
      - *par exemple : /home/florian/.ssh/devsecops.pub*

Activer une authentification multi-facteurs
- Introduction à l'authentification multi-facteurs :
  - Ce que l'on sait
    - pass...
  - Ce que l'on possède
    - carte d'accès, dongle...
  - Ce que l'on est
    - biométrie
  - Ce que l'on fait
    - analyse comportementale
  - ...
  - Dans notre exemple on utilisera un TOTP (Time-based One-Time Password) pour répondre au critère "ce que l'on a", en plus de notre clé publique/privée pour l'accès.
  - Doc Google Authenticator
    - **https://github.com/google/google-authenticator/wiki/.**
- Installation de Google Authenticator
  - Ouvrez le ficheir "ansible/3/multifactor_auth.yml
    - Contient 7 tâches, chacune ayant un job spécifique pour activer la 2FA
      - Installer le paquet libpam-google-authenticator
      - Remplacer la configuration par défaut de GoogleAuthenticator
      - Désactiver l'authentification par mot de passe pour SSH
      - Configurer PAM pour utiliser GoogleAuthenticator pour les connexions par SSH
      - Activer ChallengeResponseAuthentication
      - Activer AuthenticationMethods pour florian, ubuntu et agrant
      - Insérer une ligne indiquant "Restart SSH Server"
  - Google Authenticator est un module PAM qui permet d'activer la 2FA sur SSH.
    Il est contenu dans le paquet "libpam-google-authenticator", ainsi que tous ses fichiers de config.
  - Ouvrez le fichier "multifactor_auth.yml" et regardez la première tâche.
    - Nous utilisons le module Ansible "apt" pour installer le paquet requis.
      apt nécessite de définir le paramètre "name". "state" est réglé sur "present" pour installer le paquet. Même si c'est souvent l'état par défaut, encore une fois il vaut mieux être explicite.
      - Le module "apt" met automatiquement à jour le cache du gestionnaire de paquet avant de tenter l'installation (apt update)
- Configuration de Google Authenticator
  - Pour configurer Google Authenticator pour un utilisateur défini, on lancerait normalement la commande "google-authenticator" (qui a été installée à partir du paquet "libpam-google-authenticator").
    Cette commande crée un fichier de configuration nommé ".google_authenticator" dans le /home de l'utilisateur.
  - Nous allons créer une tâche Ansible pour copier ce fichier de config vers la VM.
    Regardez la tâche à la ligne 7 du fichier " multifactor_auth.yml"
    - Nom de la tâche
    - Module Ansible "copy"
    - Permissions : rw pour l'utilisateur seulement
- Configuration de PAM pour Google Authenticator
  - De manière assez similaire à ce que nous avons fait au chapitre précédent, nous allons de nouveau configurer PAM pour utiliser Google Authenticator pour sécuriser les connexions SSH.
    - *Fichier : "/etc/pam.d/sshd"*
  - Voir tâche ligne 15
    - Désactive l'invite de mot de passe pour SSH.
      Le module "lineinfile" localise la ligne via une regex et la commente avec un #.
    - La tâche suivante (ligne 21) configure PAM en éditant le fichier "/etc/pam.d/sshd"
      - Module lineinfile pour ajouter une ligne en fin de fichier.
      - L'option "nullok" défini cette méthode d' authentification comme optionnelle, ce qui évite de bloquer les utilisateurs tant qu'ils n' ont pas configuré la 2FA.
        Cette option devra être supprimée en prod dès que tous les utilisateurs auront activé la 2FA.
- Configuration du serveur SSH
  - Nous devons configurer SSH pour qu'il demande le code de vérification (token) à la connexion d'un utilisateur.
  - Puis nous imposerons que l'utilisateur " florian" fournisse à la fois une clé publique ET un token TOTP pour se connecter.
    On conservera l'authentification par clé publique pour les utilisateurs ubuntu et vagrant afin de garder un moyen de se connecter si un problème survenait.
  - Regardez la tâche ligne 26 :
    - Module lineinfile pour modifier une ligne dans le fichier de config du serveur SSH.
      Il localise la ligne voulue via une regex (^= début de ligne ; (yes|no) pour trouver les deux cas).
    - On a activé le prompt de saisie du token TOTP.
  - Tâche suivante :
    - Module "blockinfile" pour manipuler un bloc de texte.
      On l'utilise pour modifier plusieurs lignes d'un coup tout en conservant l'indentation du fichier.
      Le pipe est une instruction Yaml pour indiquer une chaîne de caractères sur plusieurs lignes.
    - On active enfin une action "notify" pour redémarrer le serveur SSH (voir section suivante).
- Redémarrage du serveur SSH en utilisant un Handler
  - Pour prendre en compte les modifications précédentes, il est nécessaire de redémarrer le service sshd.
    On va utiliser l'option Ansible "notify" pour le faire.
    - notify active un "handler" pour exécuter une tâche.
    - Un handler est une tâche "normale" mais elle n'est exécutée qu'une seule fois et doit posséder un nom unique à travers tout le playbook.
  - Les Handlers sont des tâches utilisées pour provoquer un arrêt ou un redémarrage d'un service en cas de changement.
  - Ouvrir le fichier "handlers/restart_sshd.yml"
    - Nom unique : Restart SSH Server
      - Doit être identique à la valeur du champ " notify" dans la tâche précédente.

Test de l'accès SSH
- Testons maintenant la connexion via Google Authenticator.
  - ssh -i ~/.ssh/devsecops -p 2222 florian@localhost
  - Si problème, vous pouvez vous connecter avec l'utilisateur vagrant et checker les logs sur la VM (/var/log/auth.log ou /var/log/syslog).
    Le serveur SSH a-t-il bien redémarré, ou y-a-t-il des erreurs dans les fichiers de config ?
  - Chacun des 10 tokens est à usage unique et est supprimé du fichier dès qu'il est utilisé pour se logger.
    Pour regénérer les tokens, lancer "vagrant provision".
  - On peut aussi utiliser une appli comme oathtool et générer un token basé sur l'heure en utilisant le secret en Base32 au début du fichier .google_authenticator.
    - *apt install oathtool*
    - *oathtool --totp --base32 " QLIUWM4UVD7E5SI6PPVZ2EGRFU"*

Provisionnement de la VM
- Pour provisionner la VM en appliquant les tâches définies jusqu'ici, nous allons devoir décommenter leurs lignes dans le playbook.
  - Ouvrir le fichier site.yml
  - Décommenter les lignes comportant " authorized_keys" et "multifactor_auth"
  - Décommenter la ligne du handler "restart_ssh. yaml"
- Nous allons maintenant automatiser la configuration de la VM en utilisant Vagrant.
  - Aller dans le dossier "vagrant"
  - Entrer la commande "vagrant provision"
  - Notez le nouveau nombre de tâches.
  - Vérifiez que tout s'est bien déroulé (aucun failed).