Projet : Audit de Conformité Automatisé et Cartographie des Risques avec la méthode EBIOS RM

Réalisé par :

Joseline YOUEGO

Table des matières

1 Présentation du projet d'audit technique automatisé

1.1 Contexte et motivation

Après avoir conçu et mis en œuvre un laboratoire d'Identity and Access Management (IAM) centré sur le provisioning automatisé, il est apparu nécessaire d'aborder la problématique complémentaire du contrôle et de l'évaluation. Dans une trajectoire professionnelle orientée vers l'audit, la gouvernance et la gestion des risques (GRC), il ne suffit pas de démontrer sa capacité à construire et exploiter une infrastructure. Il est tout aussi important de montrer que l'on sait l'auditer, interpréter les résultats et les relier à des enjeux métiers.

Dans cette perspective, ce projet a pour objectif de mettre en place une chaîne d'audit technique automatisée sur un serveur Linux, puis de traduire les constats techniques en scénarios de risques formalisés, selon une démarche inspirée d'EBIOS RM et des bonnes pratiques de l'ISO 27005.

1.2 Objectifs du projet

Le projet poursuit deux objectifs principaux.

- Mettre en œuvre un audit de conformité automatisé sur un serveur Ubuntu en s'appuyant sur le standard OpenSCAP et sur des référentiels reconnus, tels que les profils de sécurité issus des guides SCAP Security Guide (SSG) ou des benchmarks du Center for Internet Security (CIS).
- Exploiter les résultats techniques de cet audit pour construire une analyse de risques structurée, en identifiant les vulnérabilités majeures, les scénarios de menace associés, les impacts potentiels pour l'organisation, puis les mesures de remédiation correspondantes.

L'enjeu est de montrer la capacité à faire le lien entre un rapport technique automatisé et une analyse de risques orientée métier, puis à démontrer l'efficacité des mesures de renforcement mises en œuvre par un nouveau cycle d'audit.

1.3 Périmètre technique et architecture du laboratoire

Le projet s'inscrit dans la continuité du laboratoire IAM existant. Plutôt que de créer une nouvelle infrastructure, le choix a été fait de réutiliser et de spécialiser une machine virtuelle déjà en place.

L'architecture retenue est la suivante.

- Une machine virtuelle `iam-app` jouant le rôle de serveur cible. Il s'agit d'un serveur Ubuntu (20.04) hébergeant une application Flask exposée via une API HTTP. Cette machine représente un serveur applicatif critique à auditer.
- Une machine virtuelle `iam-control` jouant le rôle de poste d'audit ou de contrôle. Cette machine peut héberger les outils nécessaires à la récupération des rapports et à l'analyse des résultats, même si l'audit OpenSCAP est exécuté directement sur la machine cible.

Le référentiel de conformité utilisé est celui fourni par SCAP Security Guide pour Ubuntu 20.04. Le projet s'appuie sur le profil de sécurité standard, qui couvre un ensemble de contrôles techniques génériques (configuration SSH, permissions de fichiers, politiques de mots de passe, services actifs, etc.).

1.4 Démarche générale

La démarche retenue suit trois grandes étapes.

1.4.1 Étape 1 : Audit technique initial

Dans un premier temps, un audit automatisé est réalisé sur la machine `iam-app` avec l'outil OpenSCAP.

Concrètement, les actions suivantes sont effectuées sur la machine cible.

- Installation des composants nécessaires : bibliothèque OpenSCAP et contenu SCAP Security Guide pour Ubuntu.
- Exécution d'un scan de conformité en utilisant un profil prédéfini pour Ubuntu 20.04, par exemple le profil `standard`.
- Génération d'un rapport au format HTML et d'un fichier de résultats au format XML.

Le rapport HTML obtenu permet de visualiser le taux de conformité global du serveur ainsi que le détail des contrôles réussis ou en échec. Ce premier rapport constitue l'état initial de la machine avant toute action de renforcement.

1.4.2 Étape 2 : Analyse de risques à partir des résultats d'audit

À partir du rapport technique, plusieurs vulnérabilités significatives sont sélectionnées, en particulier celles qui apparaissent comme critiques ou ayant un impact potentiel élevé sur la sécurité du système.

Pour chacune de ces vulnérabilités, une fiche de risque est construite en s'inspirant de la méthode EBIOS RM. Chaque fiche contient notamment les éléments suivants.

- La vulnérabilité technique identifiée dans le rapport OpenSCAP, par exemple une configuration SSH non conforme.
- Le bien support concerné (par exemple le système d'exploitation du serveur applicatif).
- Le bien essentiel associé (par exemple les données manipulées par l'application ou la disponibilité du service).
- La source de menace plausible (attaquant externe, administrateur malveillant, utilisateur interne, etc.).
- Le scénario de menace décrivant la manière dont la vulnérabilité pourrait être exploitée.
- Les impacts potentiels en termes de confidentialité, d'intégrité et de disponibilité.
- Les mesures de sécurité recommandées (durcissement de la configuration, modification de paramètres, restriction d'accès, etc.).

Cette étape permet de passer d'une vision purement technique à une vision orientée risques, en reliant chaque non-conformité à des enjeux concrets pour l'organisation.

1.4.3 Étape 3 : Mise en œuvre des mesures de remédiation

Les mesures de sécurité identifiées sont ensuite appliquées sur la machine `iam-app`. Il s'agit, par exemple, de modifier la configuration du service SSH dans le fichier `/etc/ssh/sshd_config`, d'ajuster les permissions de certains fichiers sensibles ou de désactiver des services non nécessaires.

Cette phase donne lieu à la rédaction d'un script de durcissement ou, a minima, à une procédure documentée décrivant précisément les commandes exécutées et les fichiers modifiés. L'objectif est de pouvoir rejouer ces actions de manière reproductible et de conserver une traçabilité des décisions prises.

1.5 Production attendue et valorisation du projet

Au terme de ce projet, plusieurs livrables sont produits.

- Un rapport d'audit initial au format HTML généré par OpenSCAP, reflétant l'état de la machine avant durcissement.

- Une matrice d'analyse des risques, présentant pour un sous-ensemble de vulnérabilités les éléments de contexte, les scénarios de menace et les mesures de sécurité associées.

- Un script ou une procédure de durcissement détaillant les modifications de configuration appliquées.

- Un rapport d'audit final au format HTML montrant l'évolution du niveau de conformité après remédiation.

En complément, l'ensemble des fichiers techniques (scripts, extraits de configuration, matrice de risques, rapports) peut être regroupé dans un dépôt Git dédié. Cela facilite la reproductibilité du laboratoire et permet de présenter le projet de manière synthétique dans un portfolio professionnel.

Ce travail illustre la capacité à articuler un audit technique automatisé avec une démarche structurée de gestion des risques, en s'appuyant sur des outils et des référentiels reconnus. Il se situe à l'interface entre expertise technique et analyse de risques, ce qui est attendu pour un profil d'auditeur ou de consultant en cybersécurité.

2 Étape 1 : Mise en place de l'audit technique automatisé

2.1 Installation d'OpenSCAP sur la machine cible

L'audit automatisé est réalisé directement sur la machine virtuelle `iam-app`, qui joue le rôle de serveur applicatif à évaluer. La première étape consiste à installer la bibliothèque OpenSCAP et l'outil en ligne de commande `oscap` fournis par le paquet `libopenscap8` dans les dépôts Ubuntu.

L'installation est effectuée avec la commande suivante :

```
vagrant@iam-app:~$ sudo apt install libopenscap8
```

À l'issue de l'installation, l'outil `oscap` est disponible dans le système. La présence de l'exécutable et sa version sont vérifiées explicitement :

```
vagrant@iam-app:~$ which oscap
/usr/bin/oscap

vagrant@iam-app:~$ oscap -V
OpenSCAP command line tool (oscap) 1.2.16
...
```

Cette vérification confirme que la commande `oscap` est bien installée, que la version 1.2.16 est opérationnelle et que l'outil prend en charge les spécifications nécessaires (XCCDF, OVAL, CPE, etc.) pour exécuter un scan de conformité basé sur un contenu SCAP.

2.2 Premier scan de conformité avec le profil standard Ubuntu 20.04

Une fois l'outil disponible, un premier audit de conformité est réalisé sur `iam-app`. Le contenu SCAP utilisé est celui fourni par `SCAP Security Guide` pour Ubuntu 20.04, déjà présent dans le répertoire local `scap-security-guide-0.1.69/` sous la forme du fichier `ssg-ubuntu2004-ds.xml`.

Le scan est lancé avec le profil `standard` et génère à la fois un fichier de résultats au format XML et un rapport lisible au format HTML :

```
vagrant@iam-app:~$ sudo oscap xccdf eval \
  --profile xccdf_org.ssgproject.content_profile_standard \
  --results resultat.xml \
  --report rapport.html \
  scap-security-guide-0.1.69/ssg-ubuntu2004-ds.xml
```

Pendant l'exécution, `oscap` affiche pour chaque règle de sécurité évaluée le titre, l'identifiant de la règle et le résultat associé (par exemple `pass`, `fail`, `notchecked` ou `notapplicable`). Un extrait représentatif de la sortie est présenté ci-dessous :

```
Title   Ensure /home Located On Separate Partition
Rule    xccdf_org.ssgproject.content_rule_partition_for_home
Result  fail

Title   Ensure Logrotate Runs Periodically
Rule    xccdf_org.ssgproject.content_rule_ensure_logrotate_activated
Result  fail

Title   Disable SSH Root Login
Rule    xccdf_org.ssgproject.content_rule_sshd_disable_root_login
Result  fail

Title   Ensure rsyslog is Installed
Rule    xccdf_org.ssgproject.content_rule_package_rsyslog_installed
Result  pass

Title   Enable cron Service
Rule    xccdf_org.ssgproject.content_rule_service_cron_enabled
Result  pass
```

Ce premier scan met en évidence un ensemble de non-conformités typiques d'une installation par défaut : absence de partitions séparées pour certains répertoires (`/home`, `/tmp`, `/var`, `/var/log`), absence ou configuration incomplète du sous-système d'audit, paramètres `sysctl` de durcissement partiellement non conformes, et plusieurs paramètres SSH non alignés avec les recommandations (par exemple Disable SSH Root Login ou Disable SSH Access via Empty Passwords en échec).

En parallèle de l'affichage en console, `oscap` produit deux fichiers :

- `resultat.xml`, qui contient le détail structuré de l'évaluation XCCDF ;
- `rapport.html`, qui présente le résultat de l'audit sous forme de tableau de bord exploitable dans un navigateur web.

Afin de pouvoir analyser ce rapport depuis la machine hôte, le fichier HTML est copié dans le répertoire partagé `/vagrant`, monté automatiquement par Vagrant :

```
vagrant@iam-app:~$ cp rapport.html /vagrant/audit_initial_FAIL.html
```

Le fichier `audit_initial_FAIL.html` peut ainsi être ouvert dans un navigateur sur le poste de travail. Il constitue la référence de l'état initial de la machine `iam-app` avant toute action de durcissement. Les règles en échec, en particulier celles relatives à SSH et à l'audit, serviront de base de travail pour la sélection des vulnérabilités à analyser dans la matrice de risques et pour la définition des mesures de remédiation dans les sections suivantes.

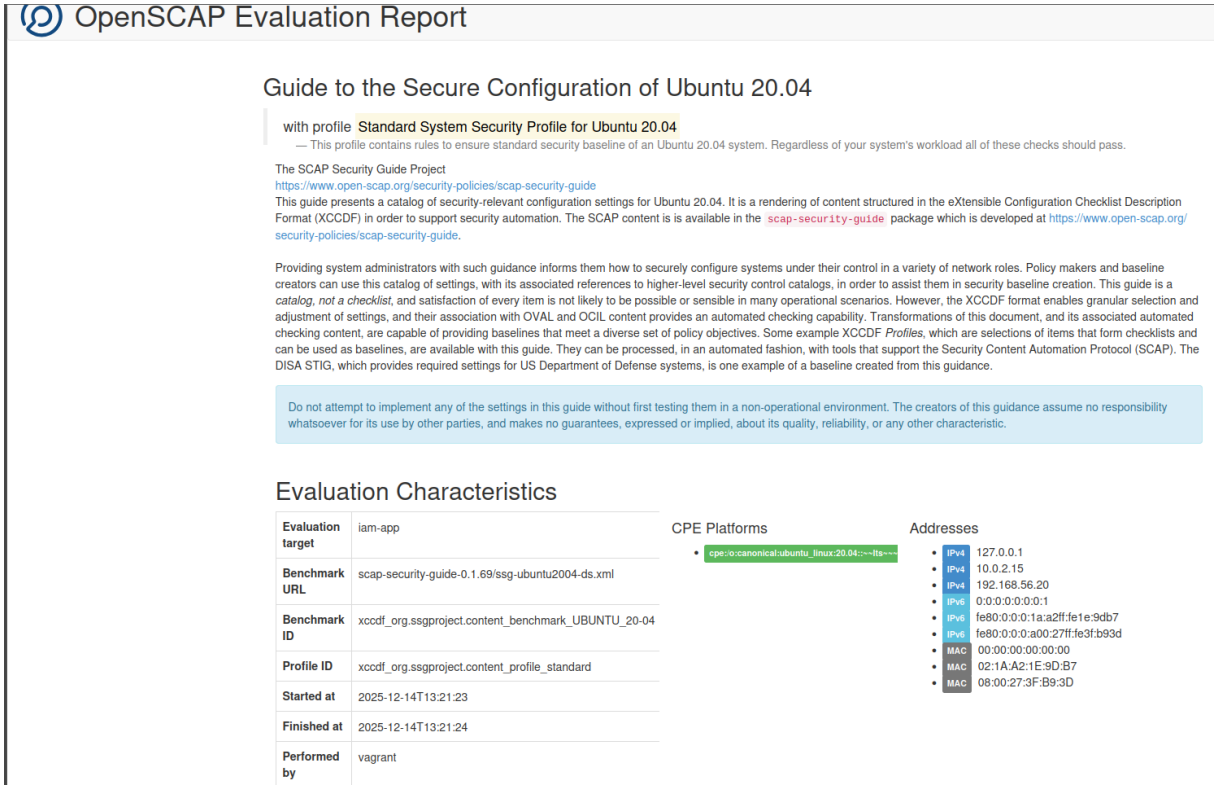

Figure 2.1: Rapport du scan - profil de la cible

3 Étape 2 : Analyse de risques basée sur le rapport OpenSCAP

À partir du rapport initial généré par OpenSCAP sur la machine `iam-app`, plusieurs non-conformités ont été identifiées. Dans le cadre de ce projet, trois vulnérabilités représentatives ont été sélectionnées afin d'illustrer la démarche de traduction d'une faiblesse technique en scénario de risque métier, selon une logique proche d'EBIOS RM. Ces vulnérabilités couvrent trois dimensions complémentaires : les accès d'administration, la protection système et l'architecture de la plateforme.

3.1 Accès d'administration SSH (compte `root`)

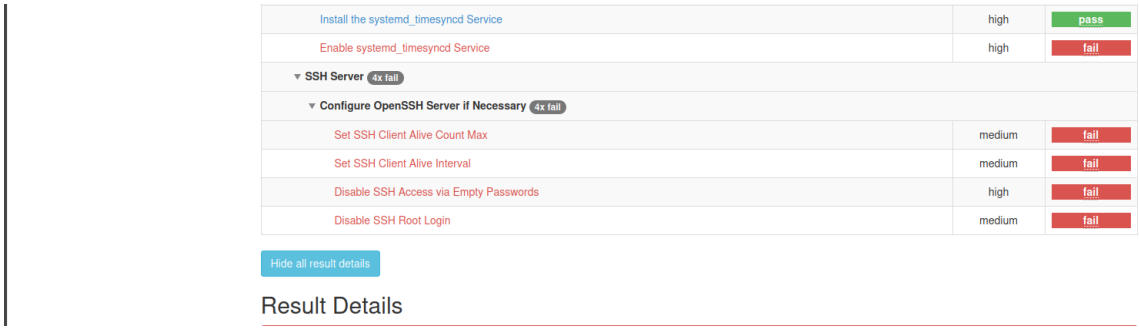

Figure 3.1: Rapport du scan - SSH

La première vulnérabilité concerne la configuration du service SSH sur le serveur applicatif. OpenSCAP signale un échec pour la règle `xccdf_org.ssgproject.content_rule_sshd_disable_root_login`, ce qui indique que le compte `root` est autorisé à se connecter directement en SSH.

Le bien support est le service d'accès distant `sshd` exposé sur le serveur. Le bien essentiel associé est l'intégrité et la confidentialité de l'ensemble des données et services hébergés sur ce serveur, notamment l'application IAM simulée.

La source de menace considérée est un attaquant externe, humain ou automatisé, qui dispose d'un accès réseau au port SSH. Le scénario de menace est le suivant : l'attaquant lance une attaque par force brute ou dictionnaire sur le compte `root`. En cas de succès, il obtient immédiatement des privilèges d'administration complets, sans étape préalable d'escalade de privilèges.

Les impacts potentiels sont élevés sur les trois dimensions classiques. En confidentialité, l'attaquant peut accéder à l'ensemble des fichiers et aux données applicatives. En intégrité, il peut modifier les configurations, altérer les journaux ou installer des portes dérobées. En disponibilité, il peut arrêter des services ou supprimer des ressources critiques, voire chiffrer le système au moyen d'un rançongiciel.

La mesure de sécurité prioritaire consiste à désactiver la connexion SSH directe du compte `root` en modifiant le fichier `/etc/ssh/sshd_config` afin de positionner le paramètre `PermitRootLogin` à `no`. Cette mesure peut être complétée par la mise en place d'une authentification par clés SSH, et par la limitation des comptes autorisés à se connecter au serveur.

3.2 Protection mémoire et randomisation de l'espace d'adressage

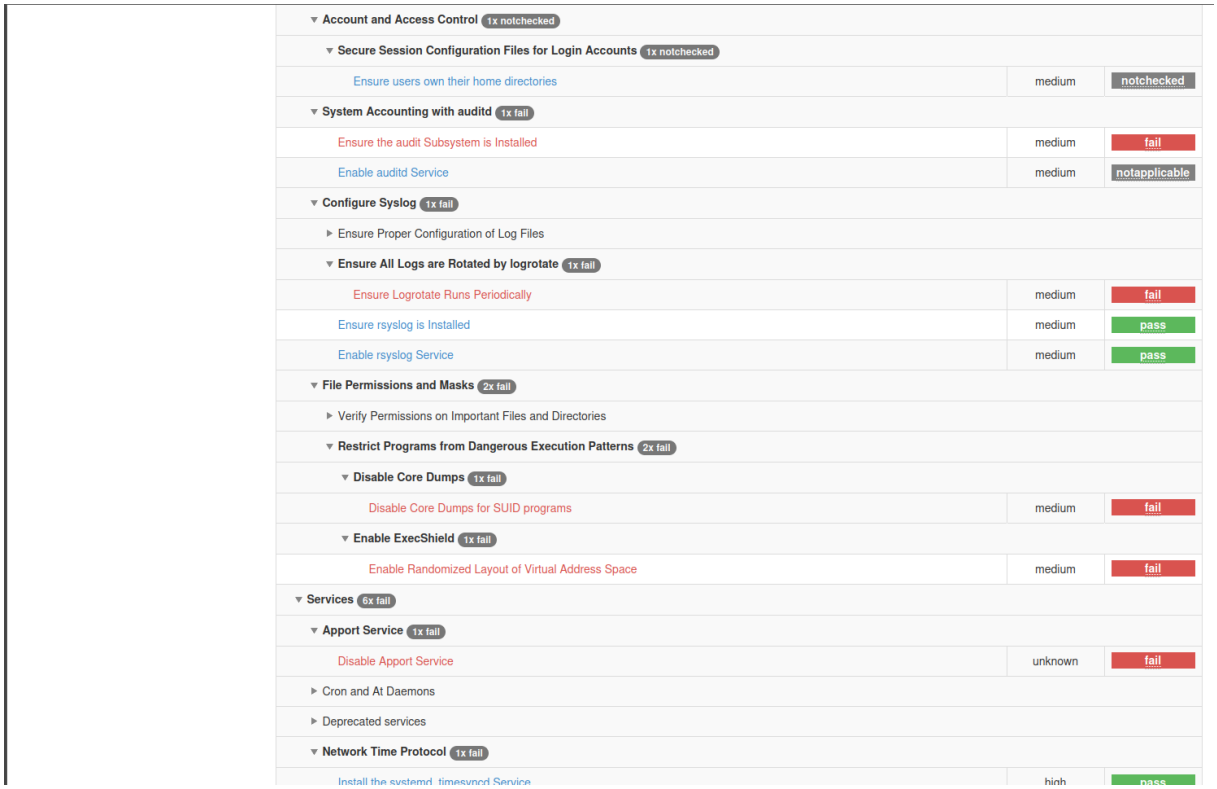

Figure 3.2: Rapport du scan - mémoire

La deuxième vulnérabilité relève de la configuration du noyau et de la protection mémoire. OpenSCAP signale un échec pour la règle `xccdf_org.ssgproject.content_rule_sysctl_kernel_randomize_va_space`, ce qui indique que la randomisation de l'espace d'adressage (ASLR) n'est pas activée ou pas configurée au niveau recommandé.

Le bien support est le noyau du système d'exploitation Ubuntu exécutant les processus de l'application Flask. Le bien essentiel est l'intégrité de l'exécution des processus applicatifs, c'est-à-dire la garantie que le code exécuté correspond bien au code légitime déployé.

La source de menace envisagée est un attaquant exploitant une vulnérabilité de type dépassement de tampon ou corruption mémoire dans l'application ou dans une bibliothèque sous-jacente. Sans ASLR, la disposition des segments mémoire est prévisible, ce qui facilite la construction d'exploits fiables.

Le scénario de menace peut être résumé ainsi : l'attaquant exploite une faille d'exécution dans l'application, injecte du code malveillant et, grâce à l'absence de randomisation, parvient à détourner le flot d'exécution du processus pour exécuter ce code. Les impacts sont principalement situés sur l'intégrité, avec la possibilité de modifier le comportement de l'application ou d'exécuter des commandes arbitraires. La disponibilité peut également être affectée en cas de plantage répété des processus.

La remédiation recommandée consiste à activer la randomisation complète de l'espace d'adressage en positionnant le paramètre `kernel.randomize_va_space` à la valeur `2` via `sysctl`, puis en rendant cette configuration persistante dans le fichier `/etc/sysctl.conf`. Cette mesure réduit significativement la probabilité de succès d'exploits mémoire classiques.

3.3 Partitionnement et saturation de l'espace disque (`/var`)

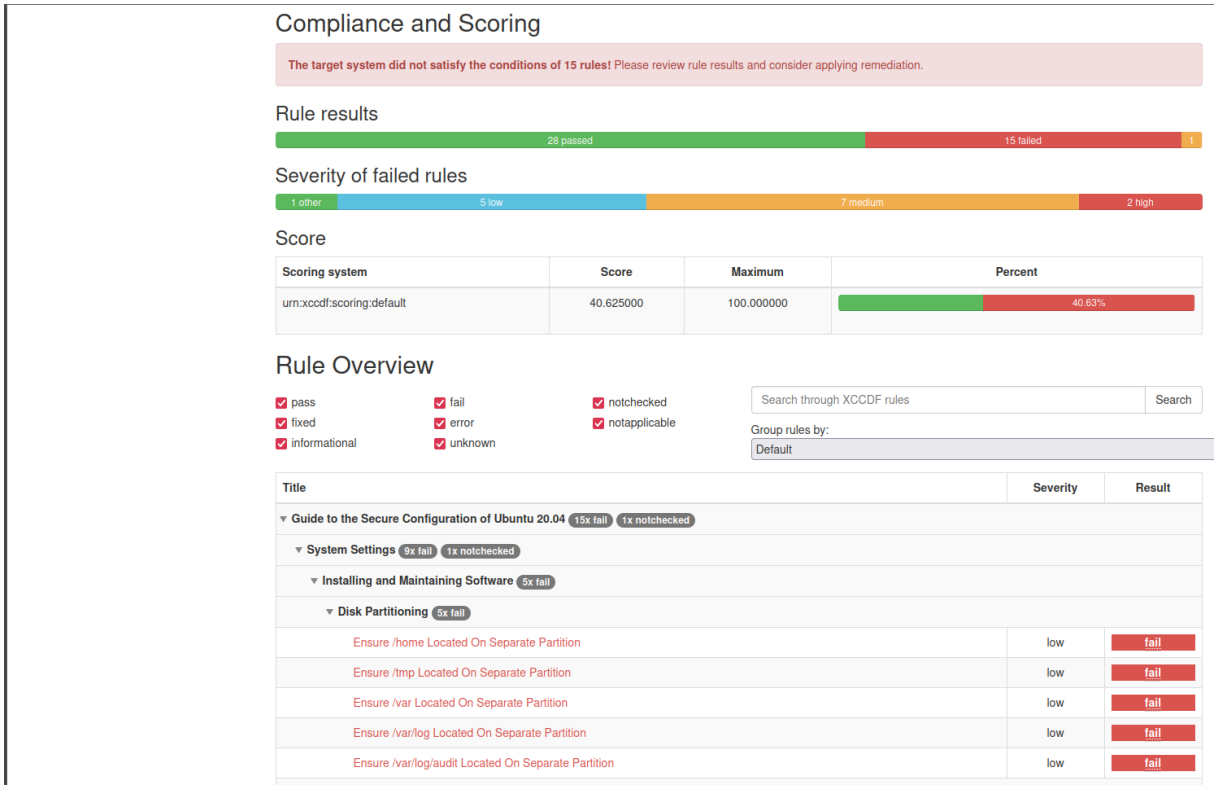

Figure 3.3: Rapport du scan - partitionnement du disque

La troisième vulnérabilité observée concerne l'architecture du système de fichiers. OpenSCAP signale un échec pour la règle `xccdf_org.ssgproject.content_rule_partition_for_var`, indiquant que le répertoire `/var` n'est pas isolé sur une partition dédiée.

Le bien support est la couche de stockage du serveur, et plus particulièrement la partition qui héberge à la fois le système et les répertoires applicatifs. Le bien essentiel associé est la disponibilité du service IAM, qui repose sur la bonne santé du système d'exploitation.

La source de menace peut être une erreur interne (processus qui génère une quantité anormale de journaux) ou un attaquant cherchant à provoquer une saturation disque par envoi massif de requêtes. Dans les deux cas, l'absence de partition dédiée pour `/var` implique que la croissance des journaux et de certains fichiers applicatifs consomme l'espace de la partition racine.

Le scénario de menace est le suivant : les fichiers sous `/var` occupent progressivement tout l'espace disponible. Lorsque le système ne peut plus écrire de fichiers temporaires ni de journaux, des dysfonctionnements majeurs apparaissent, pouvant aller jusqu'au blocage complet du système et à l'impossibilité de se connecter.

L'impact principal est la disponibilité du service, avec un risque d'interruption totale du serveur. Dans un contexte de production, ce type de situation se traduit par une indisponibilité prolongée et, potentiellement, par une perte de données récentes si des services critiques ne peuvent plus journaliser correctement leurs opérations.

La mesure de sécurité de long terme consiste à revoir le partitionnement, par exemple en plaçant `/var`, `/home` et `/tmp` sur des volumes dédiés (potentiellement gérés en LVM). Dans le cadre de ce laboratoire, une mesure compensatoire immédiate consiste à activer et durcir la rotation des journaux via `logrotate`, et à mettre en place une surveillance de l'espace disque afin de détecter rapidement toute croissance anormale.

Synthèse de l'étape 2 : Cette étape montre comment un rapport technique généré automatiquement par OpenSCAP peut être exploité pour construire des scénarios de risques structurés. Chaque vulnérabilité détectée est associée à un bien support, à un bien essentiel, à une source de menace et à un scénario exploitable, puis reliée à des mesures de sécurité concrètes. Cette démarche permet de faire le lien entre la conformité technique et la gestion des risques, en préparant la phase suivante de remédiation et de re-scan.

4 Étape 3 : Remédiation et durcissement de la configuration

Après l'étape d'analyse de risques, l'objectif est de démontrer qu'il est possible de traduire les constats techniques en actions de sécurité concrètes, reproductibles et vérifiables. Dans ce laboratoire, la remédiation ne se fait pas manuellement règle par règle, mais au moyen d'un script de durcissement, ce qui s'inscrit dans une logique d'industrialisation et d'Infrastructure as Code.

Compte tenu des résultats du premier scan OpenSCAP, deux vulnérabilités ont été traitées en priorité : la possibilité de connexion SSH directe avec le compte `root`, et l'absence de randomisation complète de l'espace d'adressage mémoire (ASLR). La non-conformité liée au partitionnement de `/var` a été conservée comme recommandation architecturale à plus long terme, car sa correction implique une refonte du schéma de partitions plutôt qu'un simple changement de configuration.

4.1 Script de durcissement `hardening.sh`

Les corrections ont été centralisées dans un script Bash `hardening.sh` exécuté sur la machine cible `iam-app`. Ce script est placé dans le répertoire personnel de l'utilisateur `vagrant` et exécuté avec les privilèges administrateur.

Listing 4.1: Script de hardening appliqué sur la VM `iam-app`

```
#!/bin/bash

# ========================================
# Script de Hardening
# Cible : Ubuntu 20.04
# Objectif : Corriger les non-conformités OpenSCAP
# ========================================

echo "[*] Démarrage du processus de durcissement..."

# --- 1. Correction SSH : Désactiver le login Root ---
echo "[+] Sécurisation du service SSH..."

# On sauvegarde le fichier de config au cas où
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak

# On utilise sed pour modifier la ligne PermitRootLogin
# On remplace "PermitRootLogin yes" (ou commenté) par "PermitRootLogin no"
```

```
sed -i 's/^#*PermitRootLogin.*/PermitRootLogin␣no/' /etc/ssh/sshd_config

# On redémarre le service pour appliquer
systemctl restart sshd

if [ $? -eq 0 ]; then
    echo "␣␣␣␣->␣Succès␣:␣Login␣Root␣désactivé."
else
    echo "␣␣␣␣->␣Erreur␣lors␣du␣redémarrage␣SSH."
fi


# --- 2. Correction Système : Activer l'ASLR (Randomize VA Space) ---
echo "[+]␣Activation␣de␣la␣protection␣mémoire␣ASLR..."

# Application immédiate (en mémoire)
sysctl -w kernel.randomize_va_space=2 > /dev/null

# Application persistante (pour le prochain redémarrage)
# On vérifie si la ligne existe déjà pour éviter les doublons
if grep -q "kernel.randomize_va_space" /etc/sysctl.conf; then
    sed -i 's/^kernel.randomize_va_space.*/kernel.randomize_va_space␣=␣2/' /etc/sysctl
    .conf
else
    echo "kernel.randomize_va_space␣=␣2" >> /etc/sysctl.conf
fi

echo "␣␣␣␣->␣Succès␣:␣ASLR␣configuré␣sur␣2␣(Full␣Randomization)."


# --- Fin ---
echo "[*]␣Durcissement␣terminé."
```

La première partie du script concerne la sécurisation du service SSH. Le fichier `/etc/ssh/sshd_config` est d'abord sauvegardé sous forme de copie de secours (`sshd_config.bak`) afin de permettre un retour arrière en cas de problème. Le script utilise ensuite l'outil `sed` pour forcer le paramètre `PermitRootLogin` à la valeur `no`, quelle que soit sa valeur précédente ou le fait qu'il soit commenté. Le service `sshd` est ensuite redémarré via `systemctl restart sshd`. Le script vérifie le code de retour de cette commande et affiche un message de succès ou d'erreur, ce qui facilite l'interprétation et le débogage.

La seconde partie du script traite l'activation de l'ASLR. Dans un premier temps, la valeur du paramètre noyau `kernel.randomize_va_space` est positionnée à `2` via la commande `sysctl -w`, ce qui applique immédiatement la configuration au système en cours d'exécution. Dans un second temps, la configuration est rendue persistante en modifiant le fichier `/etc/sysctl.conf`. Le script vérifie si une ligne contenant `kernel.randomize_va_space` existe déjà ; si c'est le cas, elle est remplacée par la valeur souhaitée, sinon une nouvelle ligne est ajoutée en fin de fichier. Cette approche évite les doublons et garantit que la configuration sera conservée après un redémarrage.

L'exécution du script se fait après lui avoir donné les droits d'exécution (`chmod +x hardening.sh`), puis via la commande `sudo ./hardening.sh`. Les messages affichés confirment le déroulement des différentes étapes, en particulier la désactivation du login

`root` et la mise en place de l'ASLR.

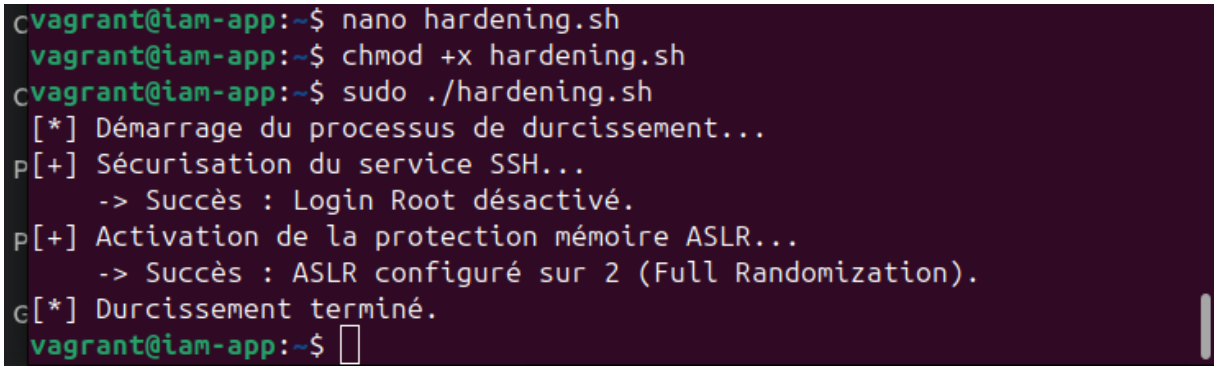

Figure 4.1: Durcissement

4.2 Re-scan OpenSCAP et comparaison avant/après

Afin de vérifier l'efficacité des mesures de durcissement, un second audit OpenSCAP a été réalisé sur la même machine `iam-app`, en réutilisant exactement le même profil que pour le scan initial. La commande d'évaluation XCCDF est identique, seules les sorties sont changées pour distinguer les résultats avant et après remédiation : les résultats XML sont stockés dans `resultat_fix.xml` et le rapport HTML dans `rapport_fix.html`. Ce dernier est ensuite copié dans le répertoire partagé `/vagrant` sous le nom `audit_final_PASS.html`, afin de pouvoir être ouvert depuis la machine hôte dans un navigateur Web.

L'analyse du rapport final montre que plusieurs règles précédemment en échec sont désormais conformes. En particulier, la règle `Disable SSH Root Login` est passée à l'état *pass*, ce qui confirme que la configuration `PermitRootLogin no` a bien été prise en compte par le service SSH.

Figure 4.2: Hardening - SSH root login

De même, la règle `Enable Randomized Layout of Virtual Address Space` apparaît désormais comme satisfaite, attestant de l'activation de l'ASLR avec le niveau de randomisation attendu.

Figure 4.3: Hardening - espace d'adressage virtuel

À l'inverse, certaines vulnérabilités structurelles, comme l'absence de partition dédiée pour `/var`, restent signalées comme non conformes. Ce résultat est cohérent avec le périmètre du script de durcissement, qui se concentre sur les corrections réalisables à chaud sans réinstallation. Cela illustre la distinction entre mesures de configuration et mesures architecturales, et justifie la formalisation de recommandations complémentaires dans la partie analyse de risques.

Cette troisième étape clôt la boucle technique du projet : un premier scan met en évidence les écarts de conformité, une analyse de risques permet de prioriser les vulnérabilités, puis un script de remédiation applique des mesures ciblées dont l'efficacité est vérifiée par un second scan. L'ensemble fournit une démonstration complète de la capacité à conduire un audit technique, à en dériver des actions de durcissement et à objectiver les progrès de sécurité par des indicateurs mesurables.

Conclusion

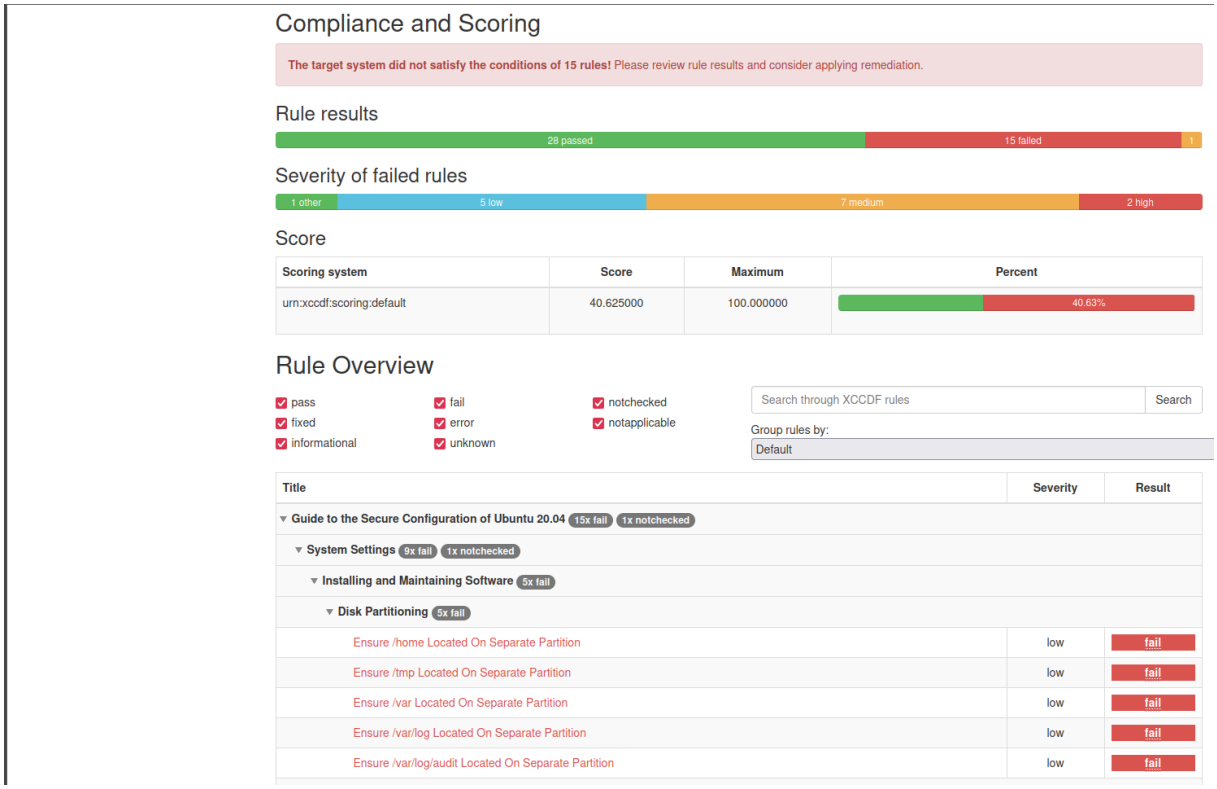

Figure 4.4: premier scan

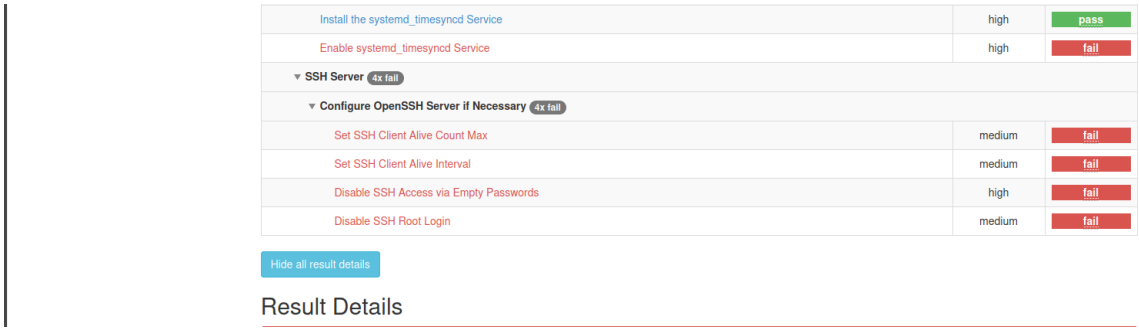

Figure 4.5: deuxieme scan

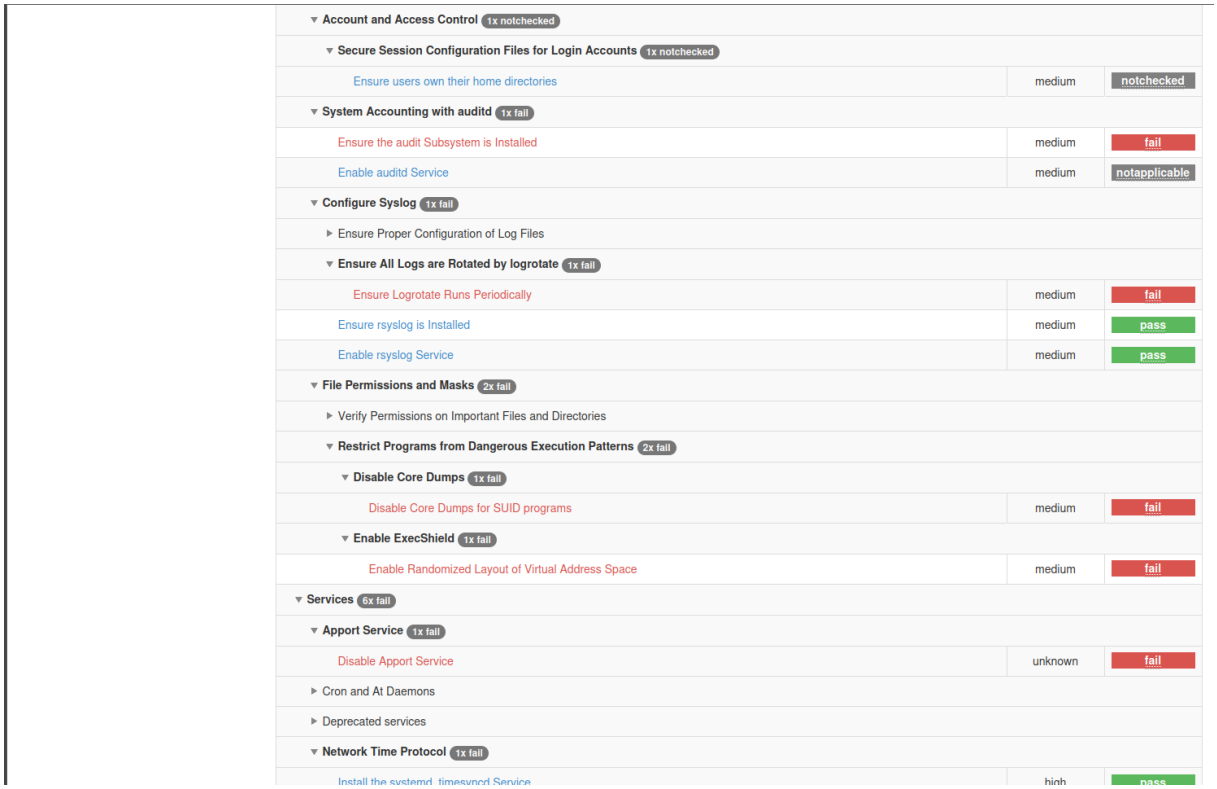

Figure 4.6: Scan - ssh

Figure 4.7: deuxieme scan - ssh

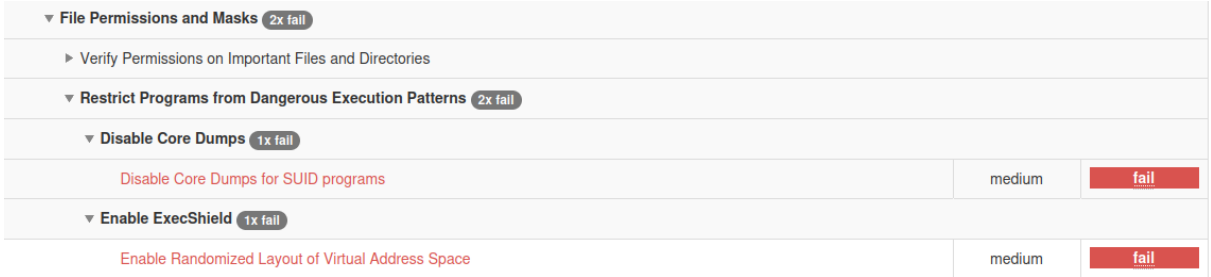

Figure 4.8: Hardening - espace d'adressage virtuel- premier scan

Figure 4.9: Hardening - espace d'adressage virtuel-deuxieme scan