Rapport Lab OS Hardening

Fait par Ugo BARBIER et Maxime HERQUE

Notre Infrastructure :

Le but de ce lab est de mettre en place une infrastructure et de la sécuriser. Cette infrastructure sera monitorée par un SIEM, dans notre cas Splunk.

Il faut dans un premier temps créer 4 VMs qui ont chacune une utilité différente. En effet, une première VM tourne sous Ubuntu 24.04 LTS et se charge d'héberger notre SIEM.

Une deuxième VM tourne sous Fedora 40 et se charge d'héberger un système de gestion de base de données : MariaDB.

La troisième VM tourne sous Debian 12 et se charge d'héberger la solution GLPI. Elle utilise la base de données hébergée sur Fedora, pour fonctionner correctement.

Et enfin, une quatrième VM tourne sous Fedora 40 et se charge d'héberger un docker stack d'Odoo, un ERP.

Détails des solutions :

Splunk est un SIEM, autrement dit un système de gestion des informations et des événements de sécurité. C'est une plateforme d'analyse de données qui permet d'agréger des données, analyser et corréler des événements, d'alerter et de visualiser les journaux système (logs) provenant de diverses sources comme des serveurs, des applications et des dispositifs réseau en temps réel. On peut facilement et rapidement avoir une vue d'ensemble sur la sécurité du parc informatique que l'on gère.

MariaDB est un système de gestion de bases de données relationnelles open source. Nous l'avons utilisé dans notre cas pour héberger la base de données de GLPI afin qu'elle se trouve sur une VM distant afin de renforcer la sécurité de notre parc.

GLPI est une solution de gestion de parc informatique, open source et modulable selon les besoins. Au travers d'un agent installé sur les assets du parc informatique, GLPI permet d'agréger des informations remontées par les assets, afin de gérer les mise à jour du parc informatique de manière centralisée par exemple. De plus, la solution

embarque un module de helpdesk, sur la base d’un service de ticketing, afin de mettre en place un service d’aide utilisateur.

Docker est une plateforme open-source qui permet de gérer et d'automatiser le déploiement d'applications dans des conteneurs isolés et portables. Elle permet de créer, tester et déployer des applications plus rapidement.

Odoo est un progiciel de gestion intégrée (ERP), mais peut aussi être utilisé comme un CMS (pour créer des sites web rapidement et facilement, à la manière de wordpress) ou comme une interface dynamique pour entreprise, avec un livechat, et un système de mail.

Objectifs

Principaux:

- Mettre en place un SIEM
- Mettre en place un serveur de base de données
- Mettre en place une solution de gestion de parc informatique
- Mettre en place un docker stack d’un ERP
- Hardening de serveur

Secondaires:

- Hardening de dockers
- Mettre en place plusieurs dashboards sur le SIEM
- Mettre en place des règles de mise à jour automatique sur GLPI
- Mettre en place un service de ticketing sur GLPI
- Mettre en place une journalisation des logs du SIEM

La séparation entre objectifs principaux et secondaires est uniquement due à la taille de notre groupe. En effet n’étant que deux, l’organisation était certes plus facile, mais les tâches à accomplir plus nombreuses. En prenant en compte les difficultés rencontrées, cela a limité les éléments que l’on a pu compléter.

Serveur Ubuntu - Splunk

Pour l'installation de Splunk, il faut dans un premier temps aller sur le site de Splunk, se créer un compte afin de récupérer le lien qui nous permettra d'installer la solution via ligne de commande.

Dépendances nécessaires

- WGET : pour la récupération des logiciels via un lien
- DPKG : pour installer les paquets Debian
- Firewalld : pour autoriser les ports nécessaire
- Splunk : logiciel pour regrouper les logs
- Splunk Forwarder : pour récupérer les logs et les envoyer à Splunk

Création / modification de configuration

Une fois l'installation du logiciel Splunk effectuée, il faut le configurer :
Pour cela, il faut créer le fichier 'inputs.conf' qui se situe dans le répertoire "/opt/splunk/etc/system/local" et mettre les éléments suivant :

Ces lignes nous permettent de mettre Splunk sur le port 9997 et de l'activer.

Dans un second temps, il faut désactiver le pare-feu de base (ufw) et installer le pare-feu 'firewalld'. Ce choix est purement personnel car nous avons appris en premier à configurer firewalld plutôt que ufw.

```
ugo@MaxUgoSplunk:/etc$ sudo ufw disable
ugo@MaxUgoSplunk:/etc$ sudo apt install firewalld
```

Il faut maintenant ouvrir le port de Splunk (9997) sur le pare-feu afin de laisser passer le trafic et que nous puissions avoir accès à l'interface Splunk :

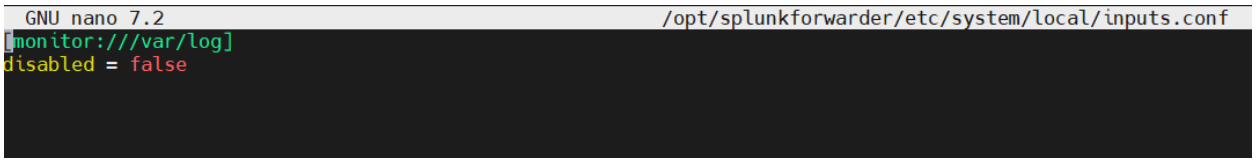
```
ugo@MaxUgoSplunk:/etc$ sudo systemctl start firewalld
ugo@MaxUgoSplunk:/etc$ sudo systemctl enable firewalld
```

```
ugo@MaxUgoSplunk:/etc$ sudo firewall-cmd --permanent
--add-port=9997/tcp
ugo@MaxUgoSplunk:/etc$ sudo firewall-cmd --reload
```

Déploiement sonde Splunk

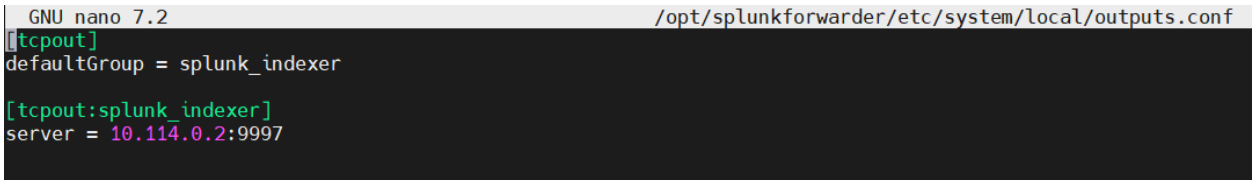
Afin de collecter les logs des différentes machines, il faut installer une sonde splunk sur chaque machine. Pour ce faire, nous allons utiliser ‘Splunk Forwarder’. Il faut donc l’installer et le configurer :
L’installation se fait de la même façon que le logiciel Splunk, c'est-à-dire avec la commande ‘wget’. Une fois la récupération du paquet réussie, il faut l’installer grâce à la commande ‘dpkg’. L’agent splunk forwarder est maintenant installé, il nous faut le configurer et la récolte des logs sera opérationnelle.
Pour ce faire, il faut modifier les fichiers de configuration ‘inputs.conf’ et ‘outputs.conf’ se situant dans le répertoire ‘/opt/splunkforwarder/etc/system/local’.

Fichier de configuration des inputs :



Ces lignes nous permettent de récolter les données se situant dans ‘/var/log’ et d’activer l’agent.

Fichier de configuration des outputs :



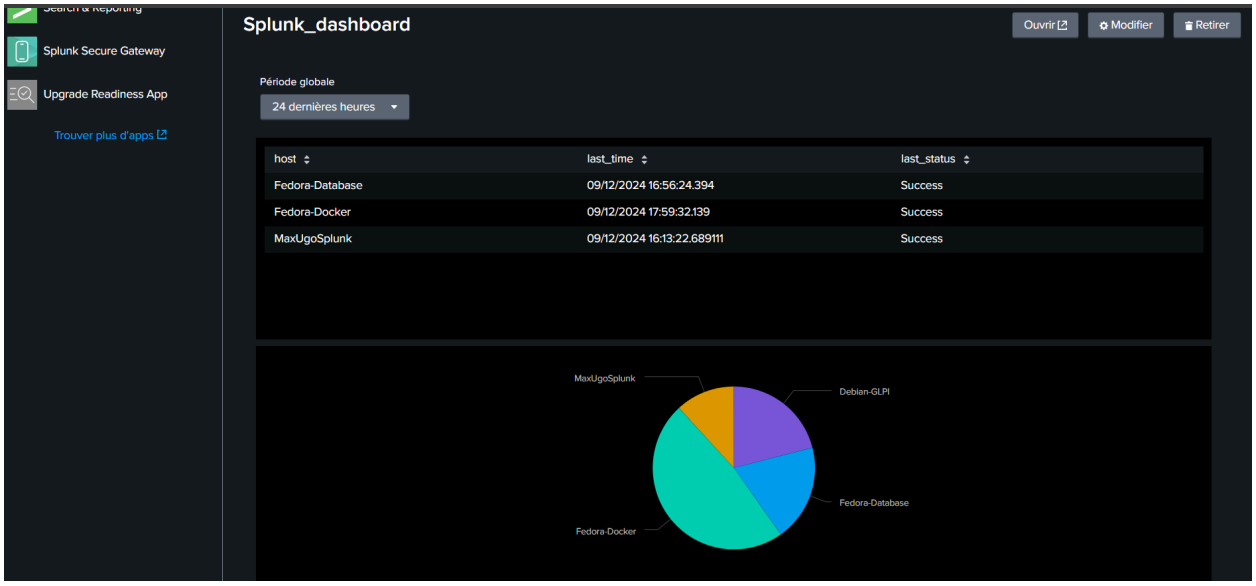
Ces lignes servent à indiquer sur quel serveur il faut envoyer les logs et via quel port.

Une fois ces déploiements effectués sur tous les serveurs, il faut redémarrer l’agent Splunk Forwarder pour que les modifications s’appliquent.

Création de dashboards

Afin d’avoir un aperçu rapide de ce qu’il se passe avec les logs, via l’interface Splunk, il faut créer des tableaux de bord. Pour ce faire, il faut dans un premier temps effectuer une recherche afin de capturer les informations souhaitées. Une fois fait, il faut

enregistrer cette recherche en tant que tableau de bord et mettre ce dernier en tableau de bord principal.



Journalisation des logs Splunk

Nous avons mis en place un script afin de sauvegarder chaques jours les logs splunk:

```
#!/bin/bash
TIME=`date +%b-%d-%y`
FILENAME=splunk-configs-backup-$TIME.tar.gz
SRCDIR=/opt/splunk/etc
DESDIR=/backup_splunk
tar -cpzf $DESDIR/$FILENAME $SRCDIR
```

Ensuite, nous avons automatisé la sauvegarde par une tâche cron, tout les jours à quatre heure du matin:

```
00 04 * * * /bin/bash /backupsplunk.sh
```

Serveur Fedora - MariaDB

Pour la création de notre base de données, nous avons besoin d'un système qui gère cette dernière. Nous avons donc choisi mariadb car elle est compatible avec notre solution GLPI.

Dépendances nécessaires

- mariadb_server & mariadb : pour la création de la base de données

Création / modification de configuration

Une fois l'installation de mariadb réussie (grâce à la commande suivante : `sudo dnf install mariadb-server mariadb`), il faut configurer ce système et créer la base de données qui servira à stocker toutes les données de GLPI.
Création de la base de données :

`sudo mysql -u root -p`
Cette commande lance mysql en root afin de configurer et créer la base de données

```
MariaDB [(none)]> CREATE DATABASE glpi_db CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci; #creation database glpi

MariaDB [(none)]> CREATE USER 'maxogu'@'%' IDENTIFIED BY
'YouWillNotKnowMySECUREPassword'; #creation username/password
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glpi_db.* TO 'maxogu'@'%';
MariaDB [(none)]> FLUSH PRIVILEGES; #increasing privileges for maxogu
```

Avec ces commandes, nous avons créé une base de données ainsi qu'un utilisateur qui a les droits pour l'interroger.

Afin de terminer la configuration de la base de données, il faut modifier le fichier de configuration de mariadb pour qu'elle autorise les connexions. Il faut donc rajouter cette ligne :
`bind-address=0.0.0.0`
Une fois la modification effectuée, il faut redémarrer le service mariadb :
`sudo systemctl restart mariadb`

Serveur Debian - GLPI

Dépendances nécessaires

- docker : pour l'installation de GLPI dans un conteneur
- snap : pour l'installation de l'agent GLPI sur tous les serveurs
- firewalld : pour avoir un pare-feu

Création / modification de configuration

Pour l'installation de docker, nous avons suivi le mode opératoire fourni directement sur le site de docker : ici
Une fois cette installation réussie, nous créons un fichier 'docker-compose.yml' afin d'exécuter plusieurs conditions en une seule fois :

```
  GNU nano 7.2                                                docker-compose.yaml
version: '3.7'

services:
  glpi:
    image: diouxx/glpi
    container_name: glpi
    ports:
      - "80:80"
    environment:
      - GLPI_DB_HOST=10.114.0.3  # IP de la VM Fedora
      - GLPI_DB_PORT=3306
      - GLPI_DB_NAME=glpi_db
      - GLPI_DB_USER=maxogu
      - GLPI_DB_PASSWORD=azerty
    volumes:
      - glpi-data:/var/www/html
    networks:
      - glpi-net

volumes:
  glpi-data:

networks:
  glpi-net:
    driver: bridge
```

Une fois le docker compose créé, il faut installer firewalld et le configurer afin de laisser la connexion passée sur le port 3306 (le port par défaut de mariadb) :

```
root@Debian-GLPI:/home/maxogu# sudo apt install firewalld
root@Debian-GLPI:/home/maxogu# sudo systemctl start firewalld
root@Debian-GLPI:/home/maxogu# sudo systemctl enable firewalld

root@Debian-GLPI:/home/maxogu# sudo firewall-cmd --permanent
```

```
--add-port=3306/tcp #authorization trafic port 3306 (mariaDB)
root@Debian-GLPI:/home/maxogu# sudo firewall-cmd --reload
```

Une fois ces configurations faites, il nous reste plus qu'à lancer le docker à l'aide de la commande suivante :

```
sudo docker compose up -d #starting docker glpi
```

La solution GLPI est maintenant installée et accessible à l'adresse IP publique du serveur.

Il faut donc se rendre sur l'adresse de notre serveur GLPI via un navigateur internet afin de terminer la configuration.
Une fois arrivé sur l'interface de GLPI, il nous est demandé d'installer ou de mettre à jour, il faut donc installer GLPI. Une fois installée, lors de la configuration, il nous est demandé de rentrer l'adresse ip de notre base de données, nous mettons donc l'adresse IP privée de notre base de données, puis nous rentrons ensuite le nom d'utilisateur et le mot de passe créés lors de la configuration de cette base de données GLPI.
GLPI est maintenant opérationnel et il nous reste plus qu'à installer les agents GLPI sur les serveurs voulu pour les faire remonter dans la solution.

Déploiement agent GLPI

Pour installer l'agent GLPI sur Debian et Ubuntu on passe par l'utilisation de snap.
On clone récupère dans un premier temps, via github, en utilisant wget, l'exécutable de l'agent GLPI.
On exécute ensuite l'agent et on défini le serveur auxquels il doit remonter les informations:

```
snap install --classic --dangerous GLPI-Agent-1.10_amd64.snap
snap set glpi-agent server=http://my-glpi-server/
```

Une fois ces deux étapes de faites, il ne reste qu'à forcer la remontée d'informations via la commande:

```
Glpi-agent
```

Désormais, les VM Ubuntu et Debian sont recensées dans le parc informatique géré par GLPI.

Serveur Fedora - Docker Stack Odoo

Dépendances nécessaires

- Docker compose

Création / modification de configuration

Dans un premier temps, il faut créer un répertoire odoo/ puis créer le docker-compose.yml à l'intérieur de ce même répertoire.
On doit ensuite éditer le fichier docker-compose.yml de la sorte:

```yaml
version: '3'
services:
  odoo:
    image: odoo:15.0
    env_file: .env
    depends_on:
      - postgres
    ports:
      - "8069:8069"
    volumes:
      - data:/var/lib/odoo
  postgres:
    image: postgres:13
    env_file: .env
    volumes:
      - db:/var/lib/postgresql/data/pgdata

volumes:
  data:
  db:
```

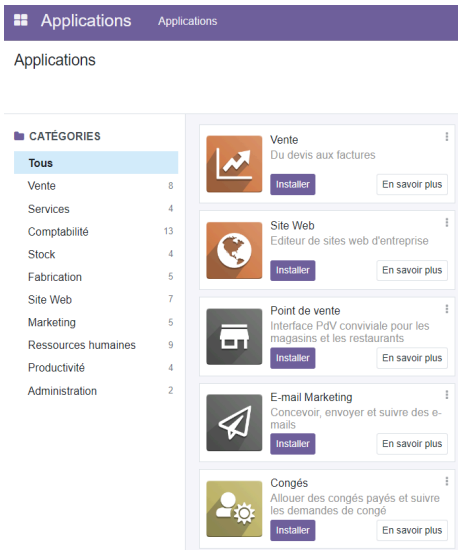
On peut constater plusieurs choses:

- On ouvre le port 8069 pour l'écoute et l'envoie de paquets
- On utilise une image odoo pré-intégrée

Il faut ensuite, éditer le fichier .env, toujours dans le répertoire /odoo:

```
# postgresql environment variables
POSTGRES_DB=postgres
POSTGRES_PASSWORD=Azerty123456789/
POSTGRES_USER=odoo
PGDATA=/var/lib/postgresql/data/pgdata

# odoo environment variables
HOST=postgres
USER=odoo
PASSWORD=Azerty123456789/
```

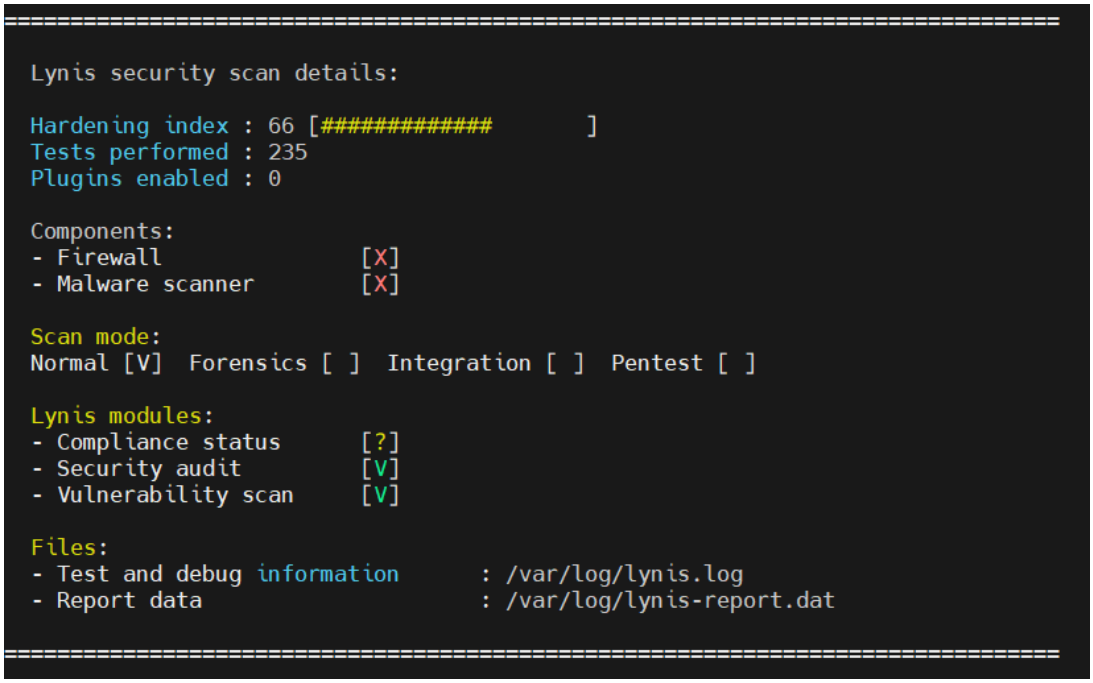
On exécute le docker stack, et l’ERP est désormais up:



Hardening

Audit initial des systèmes avec Lynis

Dans un premier temps, une fois tous les services installés, et fonctionnels, nous avons audité avec Lynis chacune des machines. Leur score de base était le même sur chacune d'entre elle:



Etapes du hardening des machines

Voici une liste des éléments de hardening mis en place:

- Mise à jour du système
- Configuration de firewall et mise en place du port 2322 en remplacement du port ssh
- Hardening des configurations ssh (tcp forwarding, max auth, max tries..)
- Création d'un utilisateur avec des droits super utilisateur afin de ne pas utiliser 'root'
- Suppression de la connexion root en ssh
- Suppression de la connexion avec mot de passe en ssh
- Connexion avec clé en ssh
- Hardening du kernel
- Permission sur le répertoire cron
- Configuration sur les règles de mot de passe (longueur, difficulté, durée de vie, non réutilisation etc..)
- Modification des hash de mot de passe

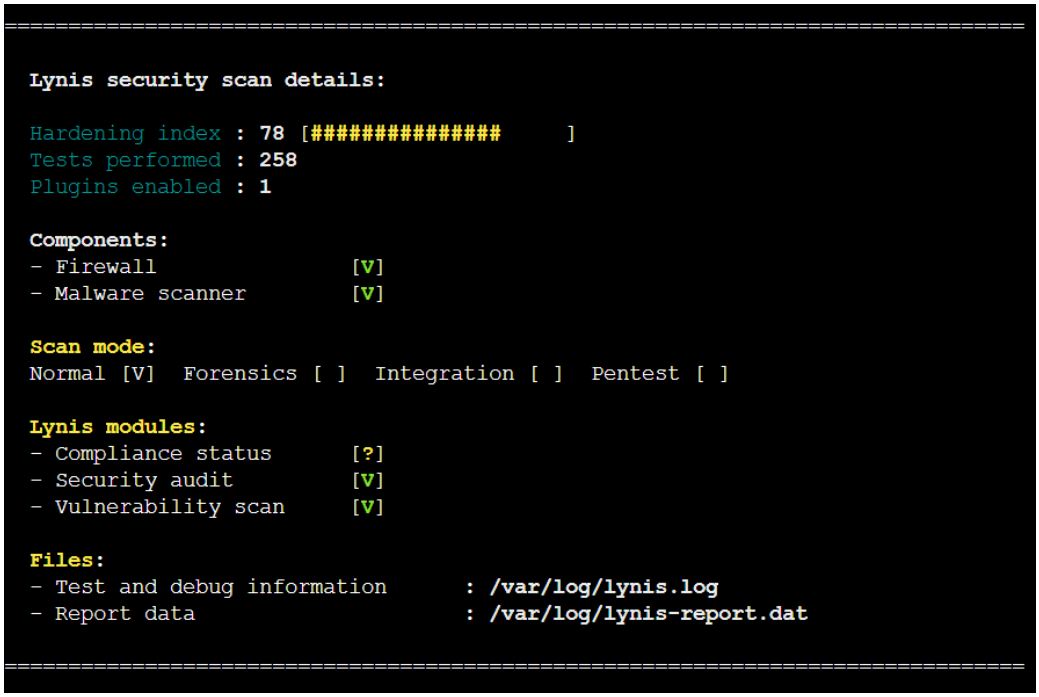
- Mise à jour automatique
- Limites à l'exécution de compilateurs
- Ajouter une “legal banner”
- Mise en place et configuration de fail2ban
- Mise en place de SELinux
- Gestion de la durée de vie et de l’expiration des comptes
- Bannissement des usb, uas, thunderbolt et firewire
- Suppression de la connexion aux base de données aux comptes root

Chacune des machines à reçu le même traitement, mais chacune n'a pas admis le même nombre de mesure de préventions. Il n’en reste pas moins que dans le temps que l’on avait, nous n’aurions pas pu faire mieux.

Audit post-hardening des machines

Une fois les machines hardennées on à repasser un audit sur chacune des machines.

La machine Ubuntu pour Splunk:



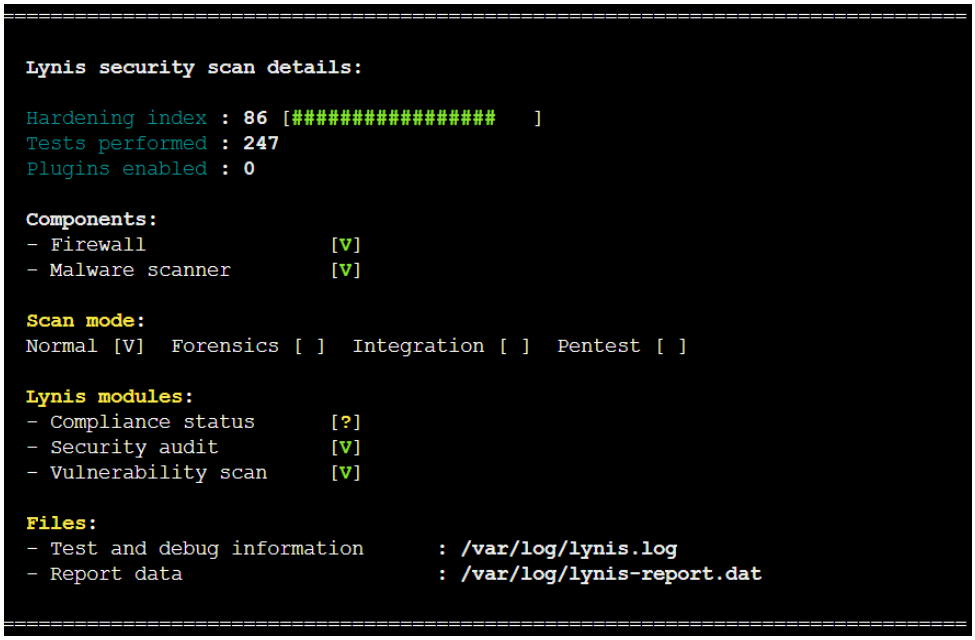
La machine Fedora pour Odoo:

```
================================================================================

  Lynis security scan details:

  Hardening index : 86 [#################   ]
  Tests performed : 261
  Plugins enabled : 0

  Components:
  - Firewall               [V]
  - Malware scanner        [V]

  Scan mode:
  Normal [V]  Forensics [ ]  Integration [ ]  Pentest [ ]

  Lynis modules:
  - Compliance status      [?]
  - Security audit         [V]
  - Vulnerability scan     [V]

  Files:
  - Test and debug information      : /var/log/lynis.log
  - Report data                     : /var/log/lynis-report.dat

================================================================================
```

La machine Debian pour GLPI:

```
================================================================================

  Lynis security scan details:

  Hardening index : 79 [###############     ]
  Tests performed : 250
  Plugins enabled : 1

  Components:
  - Firewall               [V]
  - Malware scanner        [X]

  Scan mode:
  Normal [V]  Forensics [ ]  Integration [ ]  Pentest [ ]

  Lynis modules:
  - Compliance status      [?]
  - Security audit         [V]
  - Vulnerability scan     [V]

  Files:
  - Test and debug information      : /var/log/lynis.log
  - Report data                     : /var/log/lynis-report.dat

================================================================================
```

La machine Fedora pour MariaDB:



On constate que la sécurité sur les machines hébergeant les services a été considérablement augmentée. En effet, ce n'est pas le maximum faisable, mais dans le temps impartis et avec les ressources disponibles, il aurait été difficile de faire mieux

Conclusion

Les objectifs que nous n'avons pas pu réaliser sont le hardening des Docker, et la mise en place d'un service de ticketing et de mise à jour automatique de parc informatique sur GLPI. Notre groupe étant trop peu nombreux pour réaliser toutes les tâches de manière efficace. Nous avons donc privilégié des éléments fonctionnels à la diversité bâclée.