scriting

Astier Guillaume

07/01/2026

Contents

Introduction

Linux est un système d'exploitation gratuit et open source utilisé par les utilisateurs avancés d'ordinateurs et les experts en cybersécurité. Un système d'exploitation est le logiciel qui contrôle les fonctions de base d'un ordinateur, telles que la gestion de sa mémoire, de son traitement et de son stockage.

Linux est populaire parmi les utilisateurs avancés d'ordinateurs et les experts en cybersécurité pour plusieurs raisons :

- Open source : Linux est open source, ce qui signifie que son code source est librement disponible pour que chacun puisse le consulter, le modifier ou le distribuer. Cela permet aux utilisateurs de personnaliser plus facilement le système d'exploitation en fonction de leurs besoins spécifiques et aux développeurs de travailler ensemble pour améliorer sa sécurité et ses performances.

- Sécurité : Linux est considéré comme plus sûr que d'autres systèmes d'exploitation, tels que Windows, car il est moins sensible aux virus et autres formes de logiciels malveillants. Cela en fait une option intéressante pour les experts en cybersécurité qui doivent protéger les informations et les réseaux sensibles contre les cybermenaces.

- Rentable : Linux est gratuit à télécharger et à utiliser, ce qui en fait une option rentable pour les particuliers et les organisations qui souhaitent économiser de l'argent sur les coûts du système d'exploitation.

- Flexibilité : Linux est hautement personnalisable et peut être adapté aux besoins spécifiques de différents utilisateurs, des particuliers aux grandes entreprises. Cela en fait une option polyvalente pour les utilisateurs avancés d'ordinateurs et les experts en cybersécurité qui ont besoin d'un système d'exploitation qui peut être adapté à leurs besoins uniques.

- Communauté : Linux dispose d'une communauté importante et active de développeurs et d'utilisateurs qui travaillent ensemble pour améliorer le système d'exploitation et s'entraident. Cela permet aux utilisateurs avancés d'ordinateurs et aux experts en cybersécurité de trouver plus facilement de l'aide et des ressources lorsqu'ils en ont besoin.

Dans l'ensemble, Linux est un système d'exploitation populaire parmi les utilisateurs avancés d'ordinateurs et les experts en cybersécurité en raison de sa nature open source, de sa sécurité, de sa rentabilité, de sa flexibilité et de sa communauté de soutien.

Cours sur la Sécurisation de Linux

Introduction

Linux est un système d'exploitation très populaire pour les serveurs, les ordinateurs personnels et les dispositifs embarqués. Grâce à sa nature ouverte et modulable, il offre de nombreuses possibilités d'adaptation. Toutefois, la sécurisation d'un système Linux est essentielle pour protéger les données et garantir une bonne gestion des risques face aux menaces.

Ce cours abordera les principales étapes et bonnes pratiques pour sécuriser un système Linux.

---

1. Mise à jour et gestion des patchs

Pourquoi est-ce important ?

Les vulnérabilités découvertes dans des logiciels ou dans le noyau Linux peuvent exposer votre système à des attaques. Maintenir votre système à jour avec les derniers patchs est crucial.

Actions recommandées :

- **Mise à jour automatique** : Configurez le système pour qu'il applique automatiquement les mises à jour critiques.
  - Sur Debian/Ubuntu :

```
sudo apt-get install unattended-upgrades
```

- **Vérification régulière des mises à jour** : Effectuez des mises à jour manuelles périodiques.
  - Sur Debian/Ubuntu :

```
sudo apt update && sudo apt upgrade
```

  - Sur Red Hat/CentOS :

```
sudo yum update
```

---

2. Gestion des utilisateurs et des groupes

Pourquoi est-ce important ?

Une mauvaise gestion des utilisateurs peut entraîner des risques liés à l'accès non autorisé.

Actions recommandées :

- **Utiliser des comptes utilisateurs spécifiques** : Créez des utilisateurs pour des tâches spécifiques au lieu d'utiliser le compte root pour tout.

```
sudo adduser utilisateur
```

- **Limiter les privilèges avec sudo** : Ne donnez des droits administratifs qu'aux utilisateurs qui en ont besoin.

```
sudo usermod -aG sudo utilisateur
```

- **Gestion des groupes** : Organisez les utilisateurs en groupes afin de mieux gérer les droits d'accès.
- **Utiliser des mots de passe forts** : Appliquez des règles strictes de mot de passe (longueur, complexité, expiration).

```
sudo apt-get install passwdqc
```

---

3. Sécurisation du SSH

Pourquoi est-ce important ?

Le SSH est un protocole utilisé pour accéder à distance à un serveur. Il est souvent ciblé par les attaquants.

Actions recommandées :

- **Désactiver l'accès root via SSH** : Interdire la connexion directe avec le compte root pour éviter les attaques par force brute.
  - Modifier `/etc/ssh/sshd_config` :

```
  PermitRootLogin no
```

- **Utiliser l'authentification par clé** : Authentifiez-vous avec des clés SSH plutôt qu'avec un mot de passe pour plus de sécurité.
  - Générer une clé SSH :

```
ssh-keygen
```

- Ajouter la clé publique dans `~/.ssh/authorized_keys` sur le serveur.

- **Changer le port SSH** : Le port par défaut pour SSH est 22, ce qui le rend une cible fréquente.
  - Modifier le fichier de configuration `/etc/ssh/sshd_config` :

```
Port 2222
```

---

4. Pare-feu et filtrage des paquets

Pourquoi est-ce important ?

Un pare-feu permet de contrôler le trafic entrant et sortant du système, réduisant ainsi la surface d'attaque.

Actions recommandées :

- **Configurer `ufw` (Uncomplicated Firewall)** : Utilisez `ufw` pour configurer facilement le pare-feu.
  - Activer et autoriser SSH :

```
sudo ufw allow 22/tcp
sudo ufw enable
```

  - Bloquer tous les autres ports :

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
```

- **Configurer `iptables` pour un contrôle fin** : Si vous avez des besoins spécifiques, `iptables` permet un filtrage plus détaillé.

- Exemple de règle basique pour autoriser uniquement certaines connexions :

```
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
sudo iptables -A INPUT -j DROP
```

---

5. Contrôle d'accès et permissions des fichiers

Pourquoi est-ce important ?

Une mauvaise gestion des permissions peut permettre à un attaquant d'accéder à des fichiers sensibles.

Actions recommandées :

- **Utilisez des permissions restrictives** : Appliquez le principe du moindre privilège. Donnez l'accès aux fichiers uniquement à ceux qui en ont besoin.
  - Vérification des permissions :

```
ls -l fichier
```

- **Utiliser SELinux ou AppArmor** : Ces systèmes de sécurité permettent de restreindre les actions des programmes au-delà des permissions classiques.
  - Sur Red Hat/CentOS, activer SELinux :

```
sudo setenforce 1
```

---

6. Sécurisation du noyau Linux

Pourquoi est-ce important ?

Un noyau vulnérable peut permettre à un attaquant de prendre le contrôle total de la machine.

Actions recommandées :

- **Désactiver les modules non utilisés** : Supprimez ou désactivez les modules noyau inutiles.
  - Liste des modules :

```
lsmod
```

  - Pour désactiver un module :

```
sudo rmmod <module>
```

- **Utiliser des options de noyau sécurisées** : Activez des fonctionnalités comme la protection contre l'exécution de code (DEP) et le renforcement de l'exécution des programmes.
  - Modifier `/etc/default/grub` pour ajouter des options de sécurité :

```
GRUB_CMDLINE_LINUX="noexec=off"
```

---

7. Surveillance et journalisation

Pourquoi est-ce important ?

Une bonne surveillance permet de détecter les activités suspectes et de réagir rapidement en cas d'incident.

Actions recommandées :

- **Activer la journalisation** : Le fichier `/var/log/auth.log` contient des informations essentielles pour la sécurité du système.
- **Utiliser des outils de surveillance** comme `fail2ban` pour bloquer automatiquement les adresses IP après plusieurs tentatives de connexion échouées.
  - Installer `fail2ban` :

```
sudo apt-get install fail2ban
```

- **Utiliser des outils comme `auditd` pour l'audit du système** : Cela permet de suivre l'activité du système et d'identifier toute action suspecte.

- Installer et activer `auditd` :

```
sudo apt-get install auditd
sudo systemctl enable auditd
```

Conclusion

La sécurisation d'un système Linux est un processus continu et itératif qui nécessite de prendre en compte plusieurs aspects : mise à jour régulière des logiciels, gestion des utilisateurs, configuration des services réseau, et surveillance active du système. En appliquant les bonnes pratiques et en restant vigilant, vous pouvez réduire significativement les risques liés à la sécurité.

Le monde de la sécurité est dynamique, alors assurez-vous de vous tenir informé des nouvelles vulnérabilités et des outils de protection disponibles pour Linux.

Cours Comparatif sur les Guides de Sécurisation Linux

Introduction

La sécurisation des systèmes Linux est un domaine crucial pour protéger les données sensibles, assurer la confidentialité et prévenir les intrusions malveillantes. De nombreux guides et méthodologies existent pour aider les administrateurs systèmes à sécuriser efficacement leurs serveurs et stations de travail. Dans ce cours, nous allons comparer quelques-uns des guides les plus populaires et les plus largement utilisés dans l'administration de Linux.

---

1. Le Guide de Sécurisation de la CIS (Center for Internet Security)

Description

Le **CIS Benchmark** est l'un des guides les plus respectés et détaillés. Il propose des recommandations de sécurisation pour différents systèmes d'exploitation, dont Linux. Le guide CIS Linux s'applique aux distributions comme Ubuntu, Debian, Red Hat, CentOS et autres.

Points forts

- **Réputation** : Le CIS est une référence largement reconnue dans l'industrie pour ses pratiques de sécurité.
- **Détails techniques** : Chaque recommandation est accompagnée de justifications et de mécanismes d'application détaillés.
- **Tests de conformité** : Il inclut des outils pour tester la conformité du système aux recommandations, facilitant ainsi la mise en œuvre et la vérification de la sécurité.
- **Niveaux de sécurité** : Le guide propose deux niveaux, un niveau "de base" pour les utilisateurs et un niveau "durci" pour des exigences de sécurité plus strictes.

Points faibles

- **Complexité** : Le guide peut être excessivement détaillé pour les administrateurs débutants ou les systèmes à faible criticité.
- **Mise en œuvre lourde** : Certaines recommandations peuvent entraîner des modifications drastiques du système, nécessitant de tester minutieusement les applications avant d'appliquer les configurations.

Idéal pour :

- Les entreprises ou les administrateurs qui ont besoin d'une sécurité maximale et d'une conformité aux normes (par exemple, PCI-DSS, HIPAA).
- Les serveurs critiques et les environnements réglementés.

---

2. Le Guide de Sécurisation de l'OWASP

Description

Le **OWASP** (Open Web Application Security Project) est une organisation mondiale dédiée à la sécurité des applications web. Bien qu'il se concentre principalement sur les applications, il offre aussi des

bonnes pratiques pour sécuriser l'infrastructure sous-jacente, y compris les systèmes d'exploitation Linux.

Points forts

- **Approche orientée vers les applications** : Très utile pour les serveurs web, les bases de données et les applications liées à des services web.
- **Pratique et facile à suivre** : Ce guide est souvent plus simple et plus direct par rapport aux autres, ce qui le rend accessible même aux administrateurs débutants.
- **Convergence avec les normes de développement sécurisé** : Il s'intègre bien avec les pratiques de développement sécurisées pour une sécurité "de bout en bout".

Points faibles

- **Moins complet sur l'infrastructure** : Le guide est principalement axé sur les aspects liés aux applications web et néglige parfois des aspects plus généraux de la sécurité système.
- **Manque de profondeur sur certaines configurations système avancées** : Si vous cherchez des configurations spécifiques au noyau ou des outils de gestion de sécurité comme SELinux, vous devrez consulter d'autres ressources.

Idéal pour :

- Les environnements où les applications web sont déployées (serveurs Apache, Nginx, etc.).
- Les entreprises souhaitant renforcer la sécurité des applications et des services web.

---

3. Le Guide Security-Enhanced Linux (SELinux)

Description

SELinux (Security-Enhanced Linux) est une architecture de sécurité qui permet de mettre en place des contrôles d'accès stricts en appliquant des politiques de sécurité au niveau du noyau. Le guide SELinux fournit des instructions détaillées pour activer et configurer SELinux sur un système Linux.

Points forts

- **Contrôle d'accès avancé** : SELinux permet un contrôle d'accès granulaire et renforcé au-delà des permissions de fichiers classiques.
- **Isolation des processus** : Avec SELinux, les processus malveillants peuvent être facilement isolés, réduisant ainsi l'impact des vulnérabilités.
- **Audits complets** : SELinux génère des journaux détaillés, ce qui est essentiel pour les audits de sécurité et la gestion des incidents.

Points faibles

- **Courbe d'apprentissage** : La configuration de SELinux peut être complexe et difficile à comprendre pour les administrateurs débutants.
- **Potentiel de conflits** : L'activation de SELinux peut parfois entraîner des conflits avec certains logiciels, nécessitant un ajustement fin des politiques de sécurité.

Idéal pour :

- Les systèmes nécessitant un contrôle d'accès granulaire.
- Les environnements où la sécurité du noyau et des processus est une priorité (par exemple, des serveurs sensibles ou des systèmes avec des données critiques).

---

4. Le Guide de Sécurisation de Debian

Description

Debian propose également son propre guide de sécurisation, adapté aux spécificités de cette distribution. Il aborde les bonnes pratiques à suivre lors de l'installation et de la configuration d'un système Debian afin de le sécuriser.

Points forts

- **Adapté à Debian** : Très utile pour les administrateurs qui utilisent spécifiquement Debian ou ses dérivés comme Ubuntu.
- **Conseils pratiques** : Le guide est centré sur des recommandations simples à mettre en œuvre, comme la gestion des paquets, la configuration de l'accès distant et la sécurisation des services.
- **Facilité d'utilisation** : Il est plus accessible que des guides plus techniques comme celui du CIS.

Points faibles

- **Moins détaillé sur certains aspects** : Pour une sécurité maximale, les administrateurs devront peut-être compléter ce guide avec d'autres ressources plus spécialisées.
- **Pas de tests de conformité** : Contrairement au guide CIS, ce guide n'inclut pas d'outils pour tester automatiquement la conformité du système aux recommandations.

Idéal pour :

- Les administrateurs débutants ou ceux qui cherchent une sécurité de base dans un environnement Debian ou Ubuntu.
- Les serveurs de taille moyenne ou ceux qui n'ont pas besoin de la complexité d'un guide comme le CIS.

---

5. Le Guide Linux Hardening de LPI

Description

Le **LPI** (Linux Professional Institute) propose une série de certifications professionnelles et fournit également un guide pratique pour renforcer la sécurité des systèmes Linux. Ce guide est souvent utilisé par ceux qui passent des certifications ou souhaitent un guide général pour la sécurisation des serveurs Linux.

Points forts

- **Approche pédagogique** : Le guide est structuré de manière pédagogique, idéale pour les étudiants ou les professionnels en formation.
- **Couvre une large gamme de sujets** : Il inclut des sujets tels que la gestion des utilisateurs, la configuration des pare-feu, et la mise en place de systèmes de détection d'intrusion (IDS).
- **Adapté à divers types d'environnements** : Convient aussi bien aux petites entreprises qu'aux entreprises ayant des besoins de sécurité de base à modérés.

Points faibles

- **Moins approfondi que d'autres guides** : Certaines recommandations ne sont pas aussi détaillées ou techniques que celles du CIS ou du SELinux.
- **Peu spécifique à une distribution** : Le guide reste relativement générique, sans se concentrer sur des distributions précises.

Idéal pour :

- Les professionnels en préparation à des certifications LPI.
- Les administrateurs qui cherchent une approche générale de la sécurité sans trop de détails techniques.

---

6. Le Guide de Sécurisation de l'ANSSI

Description

L'**ANSSI** (Agence Nationale de la Sécurité des Systèmes d'Information) est l'autorité française en matière de sécurité des systèmes d'information. Elle propose des guides de sécurisation détaillés, spécifiquement orientés vers la protection des infrastructures informatiques, notamment pour les administrations, les entreprises et les services publics. Son guide de sécurisation des systèmes d'information sous Linux s'adresse à un large public, allant des professionnels de la sécurité informatique aux administrateurs de systèmes.

Points forts

- **Normes de sécurité élevées** : Le guide de l'ANSSI est basé sur les meilleures pratiques et les normes de sécurité les plus rigoureuses. Il prend en compte les besoins de sécurité des systèmes d'information critiques.
- **Conformité aux normes françaises et européennes** : Le guide est aligné avec les normes de sécurité françaises et européennes, ce qui en fait un excellent choix pour les organisations en France ou celles soumises à des normes de sécurité strictes (par exemple, les administrations publiques, les entreprises travaillant avec des données sensibles, etc.).
- **Adaptabilité** : Le guide propose des recommandations de sécurisation qui s'adaptent aux besoins des systèmes en production, tout en garantissant un équilibre entre sécurité et performance.
- **Documentation complète** : L'ANSSI propose des guides détaillés qui couvrent un large éventail de domaines : de la sécurisation du noyau Linux à la gestion des accès utilisateurs, en passant par la sécurisation des applications et des réseaux.

Points faibles

- **Complexité pour les débutants** : Comme le guide CIS, celui de l'ANSSI est plutôt destiné aux administrateurs expérimentés. Il peut être difficile à suivre pour un débutant en sécurité Linux, notamment à cause de la profondeur technique et des configurations avancées.

- **Manque de tests automatisés** : Contrairement au guide CIS, l'ANSSI ne fournit pas de mécanismes automatisés pour vérifier la conformité des systèmes, ce qui peut rendre l'audit de sécurité plus manuel.
- **Contexte très centré sur la France** : Bien que largement applicable, certaines recommandations peuvent être plus pertinentes pour des organisations basées en France ou dans l'Union Européenne, ce qui peut limiter son applicabilité pour des utilisateurs en dehors de cette zone géographique.

Idéal pour :

- **Les entreprises et institutions publiques en France** : L'ANSSI étant une autorité publique française, son guide est particulièrement adapté pour les organisations soumises aux exigences de sécurité des administrations françaises ou des entreprises avec des données sensibles.
- **Les infrastructures critiques et les systèmes sensibles** : L'ANSSI se concentre sur les environnements qui nécessitent une sécurité renforcée, ce qui en fait un excellent choix pour la sécurisation de serveurs ou d'infrastructures de haute importance.
- **Les professionnels expérimentés** : Si vous avez déjà une bonne maîtrise des concepts de sécurité Linux et que vous souhaitez durcir vos systèmes en suivant les meilleures pratiques et normes de sécurité, ce guide sera un excellent complément.

Conclusion

Il n'existe pas de solution unique en matière de sécurisation Linux. Le choix du guide dépend largement des besoins spécifiques du système et des objectifs de sécurité.

- **Pour les environnements régulés et les exigences de sécurité strictes** : Le guide **CIS** est une excellente référence.
- **Pour les applications web et les serveurs associés** : Le guide **OWASP** se concentre davantage sur la sécurité des applications.
- **Pour un contrôle d'accès strict au niveau du noyau** : Le guide **SELinux** offre des solutions de sécurité avancées.
- **Pour les utilisateurs de Debian** : Le guide de sécurisation de **Debian** est simple et direct.
- **Pour une approche pédagogique et certifiante** : Le guide du **LPI** est adapté pour une formation générale.

En fonction de vos priorités de sécurité et de votre niveau d'expertise, vous pouvez choisir le guide qui correspond le mieux à vos besoins et à votre environnement.

Cours : Scripting Bash avec Vérification et Application d'un Guide de Sécurisation CIS

Introduction

Le scripting Bash est un outil puissant pour automatiser des tâches administratives et de gestion sur un système Linux. Cependant, lorsque ces scripts sont utilisés pour la configuration ou la gestion de la sécurité d'un système, il est essentiel d'intégrer des pratiques de sécurité robustes. Ce cours se concentrera sur la création de scripts Bash qui respectent les recommandations du **CIS (Center for Internet Security)** pour sécuriser un système Linux, et comment vérifier l'application des mesures de sécurisation via ces scripts.

L'objectif est de combiner la puissance du scripting Bash avec une approche systématique de sécurité, permettant aux administrateurs de vérifier et d'appliquer les bonnes pratiques définies par le **CIS Benchmark** pour Linux.

Objectifs du Cours

- **Comprendre le guide CIS** : Introduction aux concepts de base du guide de sécurisation CIS pour Linux.
- **Automatiser la vérification de sécurité** : Créer des scripts Bash qui automatisent la vérification des configurations de sécurité recommandées.
- **Appliquer les recommandations CIS** : Rédiger des scripts pour appliquer les bonnes pratiques de sécurisation sur un système Linux.
- **Pratiques de sécurité dans les scripts** : Comment rédiger des scripts Bash sécurisés pour éviter les vulnérabilités communes.

---

1. Introduction au Guide CIS pour Linux

Le guide **CIS Linux Benchmark** propose une série de recommandations détaillées sur la sécurisation des systèmes Linux. Il couvre des aspects comme la gestion des utilisateurs, la configuration du noyau,

les permissions des fichiers, la gestion des services réseau et bien d'autres éléments cruciaux pour renforcer la sécurité d'un serveur ou d'une station de travail.

Exemples de domaines traités par le guide CIS pour Linux :

- **Gestion des utilisateurs** : Sécurisation des comptes utilisateurs et des permissions.
- **Services réseau** : Configuration des services exposés sur le réseau (SSH, HTTP, etc.).
- **Logs et audits** : Mise en place des outils de suivi des événements et des journaux système.
- **Hardening du noyau** : Modification des paramètres du noyau pour améliorer la sécurité.

---

2. Scripting Bash pour Vérifier les Recommandations CIS

L'une des étapes cruciales de la sécurisation est de vérifier régulièrement que les configurations recommandées par le CIS sont appliquées. Cela peut être fait facilement à l'aide de scripts Bash.

Exemple de Script : Vérification de la configuration du SSH

Le CIS recommande de désactiver l'accès SSH par mot de passe et de n'autoriser que l'authentification par clé publique. Voici un script pour vérifier que cette configuration est correctement appliquée :

```
#!/bin/bash

# Vérification de la configuration SSH pour interdire l'accès par mot de passe
ssh_config_file="/etc/ssh/sshd_config"

# Vérifier si PasswordAuthentication est désactivé
grep -q "^PasswordAuthentication no" $ssh_config_file

if [ $? -eq 0 ]; then
    echo "Sécurisation SSH : Accès par mot de passe désactivé."
else
    echo "ALERTE : Accès par mot de passe encore autorisé, il faut le désactiver."
fi
```

Explication du script :

- grep : Recherche dans le fichier de configuration SSH pour vérifier si l'option PasswordAuthentication no est présente.
- $? : Vérifie le code de retour de la commande précédente (grep), qui est 0 si la commande a réussi (trouvé le texte).
- Si l'option n'est pas trouvée, le script alerte l'utilisateur pour qu'il désactive cette option manuellement.

---

3. Application des Recommandations CIS via Script Bash

Une autre étape cruciale est l'application des recommandations de sécurité proposées par le CIS. Cela permet d'automatiser le durcissement du système, ce qui est particulièrement utile pour déployer des configurations sécurisées à grande échelle. Exemple de Script : Appliquer les Permissions Correctes sur les Fichiers Systèmes

Le guide CIS recommande de restreindre les permissions d'accès sur des fichiers sensibles comme /etc/passwd ou /etc/shadow. Voici un script pour s'assurer que les permissions sont correctement définies.

```
1
2  #!/bin/bash
3
4  # Fichiers à sécuriser
5  files_to_check=("/etc/passwd" "/etc/shadow")
6
7  for file in "${files_to_check[@]}"; do
8      # Vérifier les permissions des fichiers
9      permissions=$(stat -c "%a" $file)
10
11     # Si les permissions sont incorrectes, les corriger
12     if [[ "$file" == "/etc/passwd" && "$permissions" != "644" ]]; then
13         echo "Mise à jour des permissions pour $file"
14         chmod 644 $file
15     elif [[ "$file" == "/etc/shadow" && "$permissions" != "0000" ]];
       then
16         echo "Mise à jour des permissions pour $file"
17         chmod 0000 $file
18     else
```

```
        echo "$file est déjà correctement sécurisé."
    fi
done
```

Explication du script :

- stat -c “%a” : Cette commande permet d’obtenir les permissions d’un fichier sous forme numérique.
- chmod : Applique les bonnes permissions si nécessaire. Par exemple, /etc/passwd doit avoir des permissions 644, et /etc/shadow doit être complètement inaccessible avec des permissions 0000.
- Le script applique automatiquement les corrections de permissions si celles-ci ne sont pas conformes.

---

4. Pratiques de Sécurité dans les Scripts Bash

Les scripts Bash peuvent eux-mêmes devenir une cible de vulnérabilités s’ils ne sont pas correctement sécurisés. Voici quelques bonnes pratiques pour rédiger des scripts Bash sûrs :

4.1. Vérification des entrées utilisateurs

Toujours valider et filtrer les entrées des utilisateurs avant de les utiliser dans un script, afin d’éviter les injections de commandes ou autres attaques.

```
#!/bin/bash

# Demander une entrée utilisateur
read -p "Entrez un nom d'utilisateur : " username

# Vérification que le nom d'utilisateur ne contient pas de caractères spéciaux
if [[ ! "$username" =~ ^[a-zA-Z0-9_]+$ ]]; then
    echo "Erreur : nom d'utilisateur invalide"
    exit 1
```

```
fi

echo "Nom d'utilisateur valide."
```

4.2. Utilisation de set -e pour arrêter le script en cas d'erreur

Cela permet d'arrêter l'exécution du script si une erreur survient, ce qui peut éviter des actions indésirables.

```

#!/bin/bash

set -e  # Arrêter immédiatement en cas d'erreur

# Exemple d'opération qui peut échouer
cp fichier_inexistant.txt /destination
```

4.3. Limiter les privilèges d'exécution

Ne jamais exécuter des scripts Bash avec des privilèges root si ce n'est pas nécessaire. Utilisez plutôt sudo pour des actions spécifiques.

```

#!/bin/bash

# Vérification que le script est exécuté en tant qu'utilisateur privilégié
if [[ $EUID -ne 0 ]]; then
    echo "Ce script doit être exécuté en tant que root"
    exit 1
fi
```

---

5. Vérification de la Conformité au CIS

Une fois que vous avez rédigé des scripts pour vérifier et appliquer les recommandations du guide CIS, il est essentiel de tester régulièrement la conformité du système. Vous pouvez automatiser cette

vérification à l'aide de scripts comme ceux présentés ci-dessus, mais il est également important d'utiliser des outils tiers pour valider la conformité au CIS. Utilisation d'outils de conformité CIS

Le CIS-CAT (CIS Configuration Assessment Tool) est un outil qui permet d'évaluer la conformité d'un système aux recommandations du CIS. Vous pouvez l'utiliser pour vérifier rapidement la configuration de votre système.

Cependant, nous allons dans ce cours, créér un outil qui permettra, sur la base du guide de sécurisation Centos 7 CIS de :

- Vérifier que les règles sont appliqué sur le système
- Appliquer les règles
- d'être lancé au démarrage de la machine et modifier la banière en indiquant le niveau de sécurisation

Exemple d'algo

Guide CIS

Prenons l'exemple de la toute première recommendation du dernier guide CIS de Centos 7

```
1.1.1.1 Ensure cramfs kernel module is not available (Automated)
Profile Applicability:
- Level 1 - Server
- Level 1 - Workstation
Description:
The cramfs filesystem type is a compressed read-only Linux filesystem
    embedded in
small footprint systems. A cramfs image can be used without having to
    first decompress
the image.
Rationale:
Removing support for unneeded filesystem types reduces the local attack
     surface of the
system. If this filesystem type is not needed, disable it.
```

Figure 1: Structure

Structure de l'application bash

Nous allons voir ensmeble un exemple de structure d'application bash permettant de gérér l'application et la vérification de la recommendation 1.1.1.1 (Ensure cramfs kernel module is not available)

Algo

PEM-CYBER-LINUX

Comme vue plus concernant l'algo et la structure du script nous allons mettre en place une première brique de code.

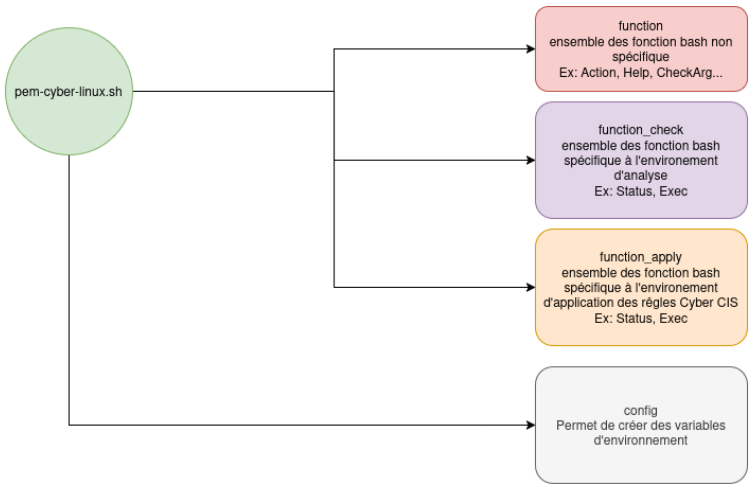

Figure 2: Algo

config

```
#!/bin/bash
DirFct=$(realpath $(dirname $0))/$(echo $(basename $0)| sed 's/.sh/.d/g
    ')/
DirCheck=$(realpath $(dirname $0))/Check
DirApply=$(realpath $(dirname $0))/Apply
#Debug='-x'
Debug=""
LogDir=/var/log/$(echo $(basename $0)\
|sed 's/.sh//g')/$(date +%H%m%d%H%M%S)
```

---

main 1

```

#!/bin/bash

## source de la configuration
source $(realpath $(dirname $0))/$(echo $(basename $0)\
| sed 's/.sh/.d/g')/config

## source des functions de base
source ${DirFct}/function

# Check if the user is root
```

```
GetRoot
# Get the argument
GetArg
# Check if all argument are present
CheckArg
```

Main 2

```
# Get specific functions
source ${DirFct}/function_${Action}

[[ $Level -eq 2 ]] && RealLevel=\* || RealLevel=1
DirAction=Dir${Action}/${Type}/${RealLevel}

[[ ! -d ${LogDir} ]] && mkdir -p ${LogDir} || true
```

Main 3

```
## MAIN
for File in $(find -type f ${DirAction} -name run.sh| sort -n)
    do
        cd $(dirname ${File})
        Test=$(basename  $(realpath .))
        bash ${Debug} ./run.sh 2>&1 | tee ${LogDir}/${Test}.log
        Res=$?
        LogTest ${Res} ${Action} ${TEST}
    done
```

---

function

```

function Usage(){
    echo "$(basename $0) [-c/-a] -l [1/2]"
    echo "-c : Check the system"
    echo "-a : Apply cyber"
    echo "-l : Level (1 or 2)"
```

```
    echo "-t : Type (desktop/server)"

    [[ ! -z $1 ]] && echo "ERROR [$1] $2"
    exit $1

}
```

```
function LogTest() {
    # LogTest ${Res} ${Action} ${TEST}
  if [[ ${1} -ne 0 ]]
    then
      echo -e "\n### ${2} - [RESULT] - ${3} : FAILED"
    else
      echo -e "\n### ${2} - [RESULT] - ${3} : SUCESS"
  fi | tee -a ${LogDir}/${3}.log
}
```

```
function GetArg(){
  OPTSTRING="hcal:t:"
  while getopts ${OPTSTRING} opt; do
    case ${opt} in
      a)  Action=Apply;;
      c)  Action=Check;;
      l)  Level=${OPTARG};;
      t)  Type=${OPTARG};;
            h)  Usage;;
      ?)  echo "Invalid option: -${OPTARG}.";  exit 1;;
    esac
  done
}
```

```

function CheckArg(){

    [[ -z ${Action} ]] && Usage 1 ": No action selected"
    [[ -z ${Level} ]] && Usage 2 ": No Level selected"
    [[ -z ${Type} ]] && Usage 3 ": No Type selected"
    [[ ${Level} -gt 2 ]] && Usage 4 ": Level need to be 1 or 2"
    true

}
```

```bash
function GetRoot() {

    if [[ $(id -u) -ne 0 ]]
        then
            Usage 5 ": You need to be root"
    fi

}
```

Function Check

Le principe de cette fonction est de vérifier le code retour de chacun des script de vérification de la sécurisation

Toutes les fonctions présente dans le fichier function_Check ne doivent pas être utiliser dans le main script mais dans les scripts présent dans le répertoire Check (run.sh)

```bash
function Check(){
        CheckRes=0
        if [[ $1 -ne 0 ]]
                then
                        CheckRes=1
        fi
}
# Export de la fonction pour être pris en compte dans les sous shell
export -f Check
```

Function Apply

Le principe de cette fonction est de vérifier que l'application de la sécurisation c'est passé correctement.

Toutes les fonctions présente dans le fichier function_Apply ne doivent pas être utiliser dans le main script mais dans les scripts présent dans le répertoire Apply (run.sh)

```bash
function Apply(){
        ExitError=$1; shift
        Step=$1; shift
```

```
4           Txt=$1;shift
5           Cmd=$@
6           eval $@
7           GetRes=$?
8           if [[ $GetRes -ne 0 ]] && [[ ExitError -eq 1 ]]
9             then
10              echo "Fatal Error on step ${Step} : $Cmd"
11              exit 10
12            else
13              [[ $GetRes -ne 0 ]] && echo "WARNING on step ${Step}" ||
                    true
14          fi
15  }
16  # Export de la fonction pour être pris en compte dans les sous shell
17  export -f Apply
```