Système, Scripts et Sécurité

Auteurs: Bouzaroura Sofiane, **Génisson** Cyril, **LEGRAND-SIMON** Trystan

Création d'une VM Debian

Installation d'une VM Debian sur VMware Workstation Pro.
Configuration:

- Processors: 2
- Memory: 4096 MG
- Hard Disk: 40 GB
- Network Adaptater: NAT

Commande de recherche avancée

```
#!/bin/bash
# created: 03/18/2024

WorkDir=$HOME
FILENAME='mon_texte.txt'
TEXT="Que la force soit avec toi"

cd $WorkDir

for k in Bureau Documents Téléchargement Vidéos Images
do
  if [ ! -d $k ]; then
    mkdir $k
  fi
  echo $TEXT > $k/$FILENAME
done
grep -rnwl $HOME -e 'force'
```

Compression et décompression de fichiers

```
#!/bin/bash
# created: 03/18/2024

WorkDir="$HOME/Documents"
FILE='Plateforme/mon_texte.txt'
TXT='Que la force soit avec toi'
TAR="Plateforme.tar.gz"

if [ ! -d $WorkDir/Plateforme ]; then
    mkdir -p $WorkDir/Plateforme
fi

cd $WorkDir
echo $TXT > $FILE

for k in {1..4}; do
    cp $FILE Plateforme/mon_texte_$k.txt
done

tar -cvzf $TAR Plateforme

cd /tmp
tar -xvzf $HOME/Documents/$TAR
```

Manipulation de texte

```
#!/bin/bash
# created: 03/18/2024

WorkDir=$HOME
CSV='users.csv'
PY='create_csv.py'

cd WorkDir
cat > $PY << EOF
#!/usr/bin/env python3
import csv

filename = 'users.csv'
liste = [
{'Nom': 'Jean', 'Age': '25 ans', 'Ville': 'Paris'},
{'Nom': 'Marie', 'Age': '30 ans', 'Ville': 'Lyon'},
{'Nom': 'Pierre', 'Age': '22 ans', 'Ville': 'Marseille'},
{'Nom': 'Sophie', 'Age': '35 ans', 'Ville': 'Toulouse'},
]

with open(filename, 'w') as f:
    fields = ['Nom', 'Age', 'Ville']
    writer = csv.DictWriter(f, fieldnames=fields)
    writer.writeheader()
    writer.writerows(liste)

EOF

chmod +x $PY
./$PY
rm -f $PY

awk '{print $1, $3}' FS="," OFS="\t" $CSV
```

Gestion des processus

```
#!/bin/bash

true <<COMMENT
  Pour recenser tous les processus actifs sur votre système :
    Commande : 'ps aux'

  Pour fermer un processus spécifique avec la commande kill, en fournissant le PID du processus que vous souhaitez terminer
    exemple : Pour tuer un processus avec le PID 1234
      'kill 1234'

  Forcer sa fermeture en utilisant la commande kill avec l'option -9 (SIGKILL).
  Cette option envoie un signal au processus pour le terminer immédiatement, sans lui donner la possibilité de se terminer
    Par exemple :
      'kill -9 1234'
COMMENT
```

Surveillance des ressources système

```
#!/bin/bash

# Nom du fichier CSV de sortie
file_system_monitoring_csv="file_system_monitoring.csv"

# En-tête du fichier CSV
echo "Timestamp,CPU(used)(%),CPU(not used)(%), Memory(%), Disk(%)" > $file_system_monitoring_csv

# Boucle infinie pour surveiller en temps réel
while true; do
  # Récupérer l'heure actuelle
  timestamp=$(date "+%d-%m-%Y %H:%M:%S")

  # Récupérer l'utilisation du CPU détaillée
  cpu_used=$(top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" | awk '{print $1}')
  cpu_not_used=$(awk "BEGIN { print 100 - $cpu_used }")
```

```
  # Récupérer l'utilisation de la mémoire
  memory_used=$(free | grep Mem | awk '{printf "%.0f", $3/$2 * 100}')

  # Récupérer l'utilisation du disque
  disk=$(df | awk '$NF=="/"{print $5}')

  echo -e "DateTime : $timestamp, CPU (used) : $cpu_used%, CPU (not used) : $cpu_not_used%, Memory : $memory_used%, Disk :
  echo -e "$timestamp, $cpu_used%, $cpu_not_used%, $memory_used%, $disk" >> $file_system_monitoring_csv

  # Attendre quelques secondes avant de répéter
  sleep 60
done
```

Scripting avancé

```
#!/bin/bash
# created: 03/18/2024


SaveDir=$HOME/save
WorkDir=$HOME/Documents
filename=Plateforme_$(date +%d-%m-%Y_%z).tar.gz

cd $WorkDir
tar -cvzf $filename Plateforme
mv $filename $SaveDir
```

```
#!/bin/bash

# Vérifier si inotify-tools est installé
if ! command -v inotifywait &> /dev/null; then
  echo "Installation de inotify-tools..."
  sudo apt-get update
  sudo apt-get install -y inotify-tools
  echo "Installation terminée."
fi

source_dir="$HOME/Documents/Plateforme"
backup_dir="$HOME/Plateforme_Backups_Save"
backup_log_file="$backup_dir/backup_log_$(date +'%m-%d-%Y_%H:%M:%S').txt"

# Créer le répertoire de sauvegarde s'il n'existe pas
mkdir -p "$backup_dir"

# Fonction pour lancer la sauvegarde
perform_backup() {
  backup_save_file="$backup_dir/backup_$(date +'%m-%d-%Y_%H:%M:%S').tar.gz"

  echo -e "\n----- SAVE PLATEFORME -----"
  echo -e "$(date +'%m-%d-%Y_%H:%M:%S') : Début de la sauvegarde de $source_dir.."
  echo "$(date +'%m-%d-%Y_%H:%M:%S') : Début de la sauvegarde de $source_dir..." >> "$backup_log_file"
  tar -czf "$backup_save_file" -C "$source_dir" . >> "$backup_log_file" 2>&1
  echo -e "$(date +'%m-%d-%Y %H:%M:%S') : Sauvegarde terminée.."
  echo "$(date +'%m-%d-%Y %H:%M:%S') : Sauvegarde terminée..." >> "$backup_log_file"
  echo "$(date +'%m-%d-%Y %H:%M:%S') : $backup_save_file" >> "$backup_dir/backup_history.txt"
  echo -e "Sauvegarde effectuée avec succès : $backup_save_file\n"
}

# Lancer une sauvegarde initiale
perform_backup
echo "$(date +'%m-%d-%Y %H:%M:%S'): Surveillance des modifications dans le répertoire source ($source_dir)..."
echo "$(date +'%m-%d-%Y %H:%M:%S'): Surveillance des modifications dans le répertoire source ($source_dir)..." >> "$backup_

# Surveillance des modifications dans le répertoire source
while event=$(inotifywait -q -r -e create,modify,delete "$source_dir"); do
  case $event in
      CREATE)
          action="Création de fichier"
          ;;
      MODIFY)
          action="Modification de fichier"
          ;;
```

```
      DELETE)
          action="Suppression du fichier"
          ;;
      *)
          # shellcheck disable=SC2034
          action="Action inconnue"
          ;;#!/bin/bash

# Fonction pour rechercher les mises à jour disponibles
search_updates() {
  echo "Recherche des mises à jour disponibles..."
  sudo apt update
}

# Fonction pour mettre à jour les logiciels
update_software() {
  echo "Mise à jour des logiciels en cours..."
  sudo apt upgrade -y
}

# Fonction Principale
main() {
  # Affichage du menu
  while true; do
    echo "Menu :"
    echo "1. Rechercher les mises à jour disponibles"
    echo "2. Mettre à jour les logiciels"
    echo "3. Quitter"


    read -r "Choisissez une option : " choix

    case $choix in
    1)
      search_updates
      ;;
    2)
      update_software
      ;;
    3)
      echo "Au revoir !"
      exit
      ;;
    *)
      echo "Option invalide. Veuillez entrer un numéro valide."
      ;;
    esac
  done
}

# Appel de la fonction principale
main
  esac
  echo "$(date +'%m-%d-%Y %H:%M:%S'): $(printf '\033[1;31m%s\033[0m' "$event") DÉTÉCTÉ !"
  echo -e "\nsauvegarde..\n"
  echo "$(date +'%m-%d-%Y %H:%M:%S'): $event détecté dans le répertoire source. Lancement de la sauvegarde..." >> "$backup_
  perform_backup
done
```

Automatisation des mises à jour logicielles

```
#!/bin/bash
# created: 03/18/2024


apt update 2>&1 /dev/null
COUNT=$(apt list --upgradable | wc -l)
if [ $COUNT -gt 1 ]; then
    read -t 10 -p "Do you want upgrade your system? [Y | n]: " answer
    if [ ! -n $answer ] | [ $answer == 'y' ] ; then
        apt upgrade -y 2>&1 /dev/null &&
        echo "System upgraded"
```

```
#!/bin/bash
# Version: 1b

# Fonction pour rechercher les mises à jour disponibles
search_updates() {
  echo "Recherche des mises à jour disponibles..."
  sudo apt update
}

# Fonction pour mettre à jour les logiciels
update_software() {
  echo "Mise à jour des logiciels en cours..."
  sudo apt upgrade -y
}

# Fonction Principale
main() {
  # Affichage du menu
  while true; do
    echo "Menu :"
    echo "1. Rechercher les mises à jour disponibles"
    echo "2. Mettre à jour les logiciels"
    echo "3. Quitter"


    read -r "Choisissez une option : " choix

    case $choix in
    1)
      search_updates
      ;;
    2)
      update_software
      ;;
    3)
      echo "Au revoir !"
      exit
      ;;
    *)
      echo "Option invalide. Veuillez entrer un numéro valide."
      ;;
    esac
  done
}

# Appel de la fonction principale
main
```

Gestion des dépendances logicielles

```
#!/bin/bash
#
# created: 03/18/2024

URL='https://deb.nodesource.com/setup_20.x'

curl -fsSL $URL | bash - &&

apt install -y git default-mysql-server apache2 php libapache2-mod-php
apt install -y php-{common,mysql,xml,xmlrpc,curl,gd,imagick,cli,dev,imap,mbstring,opcache,soap,zip,intl}
apt install -y phpmyadmin node.js
a2enmod php8.2
systemctl restart apache2
systemctl restart mariadb-server
```

```
#!/bin/bash
# Version 1b

update_upgrade_package() {
  update_command="apt update"
  upgrade_command="apt upgrade -y"
  eval "$update_command && $upgrade_command"
```

```
} # OK

# Fonction pour l'installation d'Apache ou Nginx
apache_install() {
  if ! dpkg -l | grep -q apache2; then
    apt install apache2 -y
    echo -e "Installation Apache2...\n"
  else
    echo -e "\nApache est déjà installé.\n"
  fi
} # OK

nginx_install() {
  if ! dpkg -l | grep -q nginx; then
    apt install nginx -y
    echo -e "\nInstallation Nginx...\n"
  else
    echo -e "\nNginx est déjà installé.\n"
  fi
} # OK

apache_status() {
  if dpkg -l | grep -q apache2; then
    status=$(systemctl status apache2)
    is_active=$(systemctl is-active apache2)

    echo -e "\n----- APACHE STATUS -----\n"
    echo "Menu :"
    echo "1. Afficher le statut complet d'Apache"
    echo "2. Afficher seulement l'état d'Apache"
    echo "3. Quitter"
    read -rp "Choisissez une option : " choix

    case $choix in
    1)
      echo -e "\n$status"
      ;;
    2)
      echo -e "\nÉtat : $is_active\n"
      ;;
    3)
      echo "Programme Arrêté."
      exit 0
      ;;
    *)
      echo "Option invalide. Veuillez entrer un numéro valide."
      ;;
    esac
  else
    echo "Apache n'est pas installé."
  fi
} # OK

nginx_status() {
  if dpkg -l | grep -q nginx; then
    status=$(systemctl status nginx)
    is_active=$(systemctl is-active nginx)

    echo -e "----- NGINX STATUS -----\n"
    echo "Menu :"
    echo "1. Afficher le statut complet de Nginx"
    echo "2. Afficher seulement l'état de Nginx"
    echo "3. Quitter"
    read -rp "Choisissez une option : " choix

    case $choix in
    1)
      echo -e "\n$status"
      ;;
    2)
      echo -e "\nÉtat : $is_active\n"
      ;;
    3)
      echo "Programme arrêté."
      exit 0
      ;;
    *)
      echo "Option invalide. Veuillez entrer un numéro valide."
      ;;
    esac
```

```
  else
    echo "Nginx n'est pas installé."
  fi
} # OK

web_server_install() {
  echo -e "\n"

  cols=$(tput cols)
  printf '%*s\n' "${cols}" '' | tr ' ' '-'

  echo -e "\n"
  echo -e "----- WEB SERVER -----\n"
  echo "Menu d'installation :"
  echo "1. Installer Apache"
  echo "2. Installer Nginx"
  echo "3. Quitter"
  read -rp "Choisissez une option : " web_server

  case $web_server in
  1)
    apache_install
    ;;
  2)
    nginx_install
    ;;
  3)
    echo "Programme arrêté."
    exit 0
    ;;
  *)
    echo "Choix invalide. Veuillez choisir 1 pour Apache ou 2 pour Nginx."
    install_web_server
    ;;
  esac
} # OK

# Fonctions d'installation de PHP
phpMyAdmin_install() {
  if ! dpkg -l | grep -q phpmyadmin; then
    sudo apt install phpmyadmin -y
    echo -e "Installation phpMyAdmin...\n"
  else
    echo -e "phpMyAdmin est déjà installé.\n"
  fi
} # OK

php_install() {
  if ! dpkg -l | grep -q php; then
    sudo  apt install php -y
    echo -e "Installation PHP...\n"
  else
    echo -e "PHP est déjà installé.\n"
  fi
} # OK

# Fonctions d'installation de la Base de Données
mariaDB_install() {
  if ! dpkg -l | grep -q mariadb-server; then
    apt install mariadb-server -y
    echo -e "Installation mariadb-server...\n"
  else
    echo -e "\nMariaDB est déjà installé.\n"
  fi
} # OK

mariaDB_status() {
  if dpkg -l | grep -q mariadb-server; then
    status=$(systemctl status mariadb)
    is_active=$(systemctl is-active mariadb)

    echo -e "\n----- MARIA DB STATUS -----\n"
    echo "Menu :"
    echo "1. Afficher le statut complet de MariaDB"
    echo "2. Afficher seulement l'état de MariaDB"
    echo "3. Quitter"
    read -rp "Choisissez une option : " choix

    case $choix in
    1)
```

```
      echo -e "\n$status"
      ;;
    2)
      echo -e "\nÉtat : $is_active\n"
      ;;
    3)
      echo "Programme arrêté."
      exit 0
      ;;
    *)
      echo "Option invalide. Veuillez entrer un numéro valide."
      ;;
    esac
  else
    echo "MariaDB n'est pas installé."
  fi
} # OK

mysql_install() {
  if ! dpkg -l | grep -q mysql-server; then
    apt install mysql-server -y
    echo -e "Installation MySQL...\n"
  else
    echo -e "\nMySQL est déjà installé.\n"
  fi
} # OK

mysql_status() {
  if dpkg -l | grep -q mysql-server; then
    status=$(systemctl status mysql)
    is_active=$(systemctl is-active mysql)

    echo -e "----- MYSQL STATUS -----\n"
    echo "Menu :"
    echo "1. Afficher le statut complet de MySQL"
    echo "2. Afficher seulement l'état de MySQL"
    echo "3. Quitter"
    read -rp "Choisissez une option : " choix

    case $choix in
    1)
      echo -e "\n$status"
      ;;
    2)
      echo -e "\nÉtat : $is_active\n"
      ;;
    3)
      echo "Programme arrêté."
      exit 0
      ;;
    *)
      echo "Option invalide. Veuillez entrer un numéro valide."
      ;;
    esac
  else
    echo "MySQL n'est pas installé."
  fi
} # OK

bdd_install() {
  echo -e "----- BASE DE DONNÉES -----\n"
  echo "Menu d'installation :"
  echo "1. Installer MariaDB"
  echo "2. Installer MySQL"
  echo "3. Quitter"
  read -rp "Choisissez une option : " bdd_choice

  case $bdd_choice in
  1)
    mariaDB_install
    ;;
  2)
    mysql_install
    ;;
  3)
    echo "Programme arrêté."
    exit 0
    ;;
  *)
    echo "Choix invalide. Veuillez choisir 1 pour MariaDB ou 2 pour MySQL."
```

```
    bdd_install
    ;;
  esac
} # OK

nodeJS_install() {
  if ! dpkg -l | grep -q nodejs; then
    apt install nodejs -y
    echo -e "Installation de NodeJS...\n"
  else
    echo -e "NodeJS est déjà installé.\n"
  fi
} # OK

git_install() {
  if ! dpkg -l | grep -q git; then
    apt install git -y
    echo -e "Installation de Git...\n"
  else
    echo -e "Git est déjà installé.\n"
  fi
} # OK

# Fonction Principale
main() {
  echo -e "\n"

  cols=$(tput cols)
  title="Logiciel de gestion des dépendances"
  padding=$(( (cols - ${#title}) / 2 ))
  printf '%*s%s%*s\n' $padding "" "$title" $padding ""
  printf '%*s\n' "${cols}" '' | tr ' ' '-'

  echo -e "\n"
  echo "Vous allez passer en mode root pour effectuer la mise à jour des paquets."
  echo -e "Veuillez entrer le mot de passe root lorsque vous y êtes invité."

  # `id -u` : renvoie l'identifiant de l'utilisateur sous forme de chaîne de caractères.
  # `"0"` : C'est l'identifiant de l'utilisateur root.
  # `sudo su -c` : vous demandez essentiellement à sudo de basculer vers l'utilisateur root temporairement et d'exécuter un
  # $(declare -f update_upgrade_package) : Cela exécute la commande declare -f update_upgrade_package, qui renvoie la défin
  if [ "$(id -u)" != "0" ]; then
    sudo su -c "$(declare -f update_upgrade_package); update_upgrade_package"
    web_server_install
    php_install
    bdd_install
    phpMyAdmin_install
    nodeJS_install
    git_install
  else
    sudo su -c "$(declare -f update_upgrade_package); update_upgrade_package"
    web_server_install
    php_install
    bdd_install
    phpMyAdmin_install
    nodeJS_install
    git_install
  fi

  echo "Menu :"
  echo "1. Afficher le statut de Apache"
  echo "2. Afficher le statut de Nginx"
  echo "3. Afficher le statut de MariaDB"
  echo "4. Afficher le statut de MySQL"
  echo "5. Quitter"

  read -rp "Choisissez une option : " choix

  case $choix in
  1)
    apache_status
    ;;
  2)
    nginx_status
    ;;
  3)
    mariaDB_status
    ;;
  4)
    mysql_status
```

```
    ;;
  5)
    echo "Au revoir !"
    exit 0
    ;;
  *)
    echo "Option invalide. Veuillez entrer un numéro valide."
    ;;
  esac
}

# Appel de la fonction principale
main
```

Sécuriser ses scripts

En suivant ces bonnes pratiques, nous pouvons réduire les risques de sécurité associés aux scripts shell. Nous gardons à l'esprit que la sécurité est un processus continu, et il est important de rester vigilant et de surveiller activement les nouvelles menaces et les meilleures pratiques de sécurité.

1. Validation des entrées utilisateur

Toujours valider les entrées utilisateur pour éviter les injections de code ou les erreurs potentielles. Utiliser des fonctions comme **read** pour récupérer les entrées utilisateur et effectuer une validation pour nous assurer qu'elles sont conformes à ce que que l'on attend du script.

2. Utilisation de variables

Utilisation des variables pour stocker des données sensibles et s'assurer qu'elles sont correctement protégées. Éviter d'utiliser des noms de variables prévisibles ou susceptibles d'être remplacés accidentellement par des commandes.

3. Limitation des privilèges

Éviter d'exécuter des scripts en tant qu'utilisateur **root** sauf si c'est absolument nécessaire. Si possible, définir des permissions strictes pour les fichiers et répertoires utilisés par les scripts.

4. Échappement des caractères spéciaux

Échapper les caractères spéciaux dans les données fournies par l'utilisateur pour éviter les attaques par injection de code. Nous pouvez utiliser des outils comme **sed**, **awk** ou **grep** pour nettoyer les entrées.

5. Mises à jour régulières

Assurer que nos scripts sont à jour et inclus des mécanismes de mise à jour automatique si nécessaire pour corriger les éventuelles vulnérabilités découvertes.

6. Sécurisation des fichiers de script

Limiter l'accès aux fichiers de script en définissant des permissions appropriées. S'assurer que seuls les utilisateurs autorisés peuvent les modifier ou les exécuter.

7. Journalisation des activités

Implémenter une journalisation pour enregistrer les activités des scripts.
Cela peut aider à identifier les comportements suspects ou les problèmes de sécurité potentiels.

8. Révision du code

Faire réviser notre code par des pairs pour repérer les éventuelles vulnérabilités ou erreurs de logique. Une autre paire d'yeux peut souvent repérer des problèmes que nous aurions pu manquer.

9. Utilisation de fonctions sécurisées

Utiliser des fonctions de sécurité intégrées lorsque cela est possible, par exemple: utilisation des fonctions de hachage intégrées pour le traitement des mots de passe au lieu de les stocker en texte brut.

10. Évitez l'évaluation de commandes dynamiques

Éviter l'utiliser des évaluations de commandes dynamiques comme **eval**, car elles peuvent introduire des vulnérabilités si elles ne sont pas correctement contrôlées.

Évidemment cette liste est loin d'âtre exhaustive, nous aurions pu inclure un niveau de protection contextuel comme *SELinux* pour nous assurer de la délimitation du périmètre d'action des scripts.

Utilisation d'API Web dans un script

```
#!/bin/bash

# URL de l'API à utiliser
API_URL="https://jsonplaceholder.typicode.com/todos?_limit=10"

# Chemin vers le fichier de logs
LOG_FILE="./logs/api_logs.txt"

# Chemin vers le fichier JSON
JSON_FILE="./logs/api_response.json"

# Fonction pour enregistrer les logs
log() {
  echo "$(date +'%m-%d-%Y %H:%M:%S') - $1" >> "$LOG_FILE"
}

# Fonction pour enregistrer la réponse au format JSON
save_json_response() {
  echo "$1" > "$JSON_FILE"
  log "Réponse JSON enregistrée dans $JSON_FILE :"
  cat "$JSON_FILE" >> "$LOG_FILE"
  echo "" >> "$LOG_FILE"
}

# Fonction pour gérer les erreurs
handle_error() {
  local error_message="Erreur: $1"
  echo "$error_message"
  log "$error_message"
  exit 1
}

# Vérification de l'existence du répertoire logs
if [ ! -d "./logs" ]; then
  mkdir -p "./logs" || handle_error "Impossible de créer le répertoire de logs"
fi

# Vérification de l'existence du fichier de logs
if [ ! -f "$LOG_FILE" ]; then
  touch "$LOG_FILE" || handle_error "Impossible de créer le fichier de logs"
fi

# Vérification de l'existence du fichier JSON
if [ ! -f "$JSON_FILE" ]; then
  touch "$JSON_FILE" || handle_error "Impossible de créer le fichier JSON"
fi

# Vérification de l'installation de curl
if ! command -v curl &> /dev/null; then
```

```
  handle_error "curl n'est pas installé. Veuillez l'installer pour exécuter ce script."
fi

# Fonction pour effectuer une requête à l'API
make_api_request() {
  local response
  response=$(curl -sS "$API_URL")

  echo "Response: $response"

  # Vérifiez si la réponse est vide ou non
  if [ -z "$response" ]; then
    handle_error "La réponse de l'API est vide."
  fi

  # Enregistrez la réponse dans le fichier JSON
  save_json_response "$response"
}

# Exécuter la requête à l'API
make_api_request
```