Vaultwarden Tools

- Vaultwarden Backup / Restore Scripts

Vaultwarden Backup / Restore Scripts

Das Backup-Script kann per Crontab täglich ausgeführt werden:

```
0 0 * * * /scripts/vaultwarden-daily-backup-encrypt.sh
```

> Vaultwarden läuft bei mir als Dockercontainer. Alle Dateien liegen im Vaultwarden Ordner.
>
> Backup_dir -> wo soll das Backup gespeichert werden
>
> Container -> Name des Vaultwarden Docker Containers
>
> Container_data_dir -> Speicherort der Dateien des Docker-Containers
>
> Passphrase_file -> Speicherort der Passphrase zum Verschlüsseln des Backups (Key)

#####---#####

Verschlüsseltes Backup erstellen

Anpassung bitte vornehmen: CONTAINER, CONTAINER_DATA_DIR, SQLITE_DB, PASSPHRASE_FILE

die PASSPHRASE Datei vorher anlegen und das Kennwort hinterlegen im Pfad /root/.xy/passphrase (Klartext)

vaultwarden-daily-backup-encrypt.sh

```
#!/bin/bash

# Set the script to exit immediately if any command fails
set -e
```

```
DATE=$(date +%d-%m-%Y)
BACKUP_DIR=/mnt/xyz
BACKUP_FILE=vaultwarden-$DATE.tar.gz
ENCRYPTED_BACKUP_FILE=vaultwarden-$DATE.tar.gz.gpg
CONTAINER=vaultwarden
CONTAINER_DATA_DIR=/vaultwarden-datafolder
SQLITE_DB=/vaultwarden-datafolder/db.sqlite3
PASSPHRASE_FILE=/root/.xy/passphrase

# create backups directory if it does not exist
mkdir -p $BACKUP_DIR
#mkdir -p $SQLITE_BACKUP_DIR

# Stop the container
/usr/bin/docker stop $CONTAINER

# Backup the vaultwarden data directory to the backup directory
tar -czf "$BACKUP_DIR/$BACKUP_FILE" -C "$CONTAINER_DATA_DIR" .

# Encrypt the backup file with gpg using the passphrase from the file
gpg --batch --passphrase-file "$PASSPHRASE_FILE" --symmetric --cipher-algo AES256 -o
"$BACKUP_DIR/$ENCRYPTED_BACKUP_FILE" "$BACKUP_DIR/$BACKUP_FILE"

# Optionally, remove the unencrypted backup file
rm "$BACKUP_DIR/$BACKUP_FILE"

# backup sqLite Datenbank
#sqlite3  "$SQLITE_DB" "VACUUM INTO '$SQLITE_BACKUP_DIR/$SQLITE_BACKUP_FILE'"
#sqlite3 "$SQLITE_DB" .dump > "$SQLITE_BACKUP_DIR/$SQLITE_BACKUP_FILE"

# Restart the container
/usr/bin/docker start $CONTAINER

# To delete files older than 30 days
find $BACKUP_DIR/* -mtime +30 -exec rm {} \;
```

######---######

Verschlüsseltes Backup wiederherstellen

> vaultwarden-restore-backup.sh

> Anpassungen bei folgenden Variablen vornehmen: CONTAINER, CONTAINER_DATA_DIR, SQLITE_DB, PASSPHRASE_FILE

> Das Script fragt den Passphrase ab, wenn dieser nicht im Ordner verfügbar ist

```
#!/bin/bash

# Set colors for output
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0m' # No Color

# Set the script to exit immediately if any command fails
set -e

SOURCE_DIR="/mnt/backupfolder"
PASSPHRASE_FILE="/root/.xy/passphrase1"
DEST_DIR="/vaultwarden-data"

# Ensure the destination directory exists
mkdir -p "$DEST_DIR"

echo -e
echo -e
echo -e
echo -e "${RED}!!! WARNUNG: UNBEDINGT DIE PFADE IM SCRIPT ANPASSEN; SOURCE_DIR & DEST_DIR
!!!.${NC}"
echo -e
echo -e
echo -e

# Function to prompt for passphrase with asterisks
prompt_passphrase() {
```

```
    echo -e "${YELLOW}Bitte geben Sie die Passphrase für die Entschlüsselung ein:${NC}"
    echo -n "Passphrase: "
    stty -echo  # Turn off echoing
    read PASSPHRASE  # Read input silently
    stty echo  # Turn on echoing
    echo  # Move to a new line after the password input
}

# Check if the passphrase file exists and is readable
if [ -f "$PASSPHRASE_FILE" ] && [ -r "$PASSPHRASE_FILE" ]; then
    PASSPHRASE_OPTION="--passphrase-file $PASSPHRASE_FILE"
else
    echo -e "${RED}Warnung: Die Passphrase-Datei $PASSPHRASE_FILE ist nicht vorhanden oder
nicht lesbar.${NC}"
    prompt_passphrase
    PASSPHRASE_OPTION="--passphrase '$PASSPHRASE'"
fi

# Print header for the date selection
echo -e "\n${GREEN}Wählen Sie ein Datum für das zu entpackende Archiv:${NC}\n"

# Generate numbered list of dates
DATES=($(ls "$SOURCE_DIR" | grep -oP '\d{2}-\d{2}-\d{4}' | sort | uniq))
for ((i=0; i<${#DATES[@]}; i++)); do
    echo -e "[${YELLOW}$((i+1))${NC}] ${DATES[i]}"
done

# Prompt user to select a date
while true; do
    read -p "Geben Sie die Nummer des gewünschten Datums ein: " CHOICE
    if [[ ! $CHOICE =~ ^[0-9]+$ ]]; then
        echo -e "${RED}Fehler: Bitte geben Sie eine gültige Nummer ein.${NC}"
    elif (( CHOICE < 1 || CHOICE > ${#DATES[@]} )); then
        echo -e "${RED}Fehler: Ungültige Auswahl. Bitte geben Sie eine Nummer innerhalb des
angegebenen Bereichs ein.${NC}"
    else
        DATE="${DATES[CHOICE-1]}"
        echo -e "Gewähltes Datum: ${GREEN}$DATE${NC}"
        break
    fi
```

```
done

ENCRYPTED_BACKUP_FILE="vaultwarden-$DATE.tar.gz.gpg"
DECRYPTED_BACKUP_FILE="vaultwarden-$DATE.tar.gz"

# Check if the encrypted backup file exists
if [ ! -f "$SOURCE_DIR/$ENCRYPTED_BACKUP_FILE" ]; then
    echo -e "${RED}Fehler: Die verschlüsselte Datei $SOURCE_DIR/$ENCRYPTED_BACKUP_FILE
existiert nicht.${NC}"
    exit 1
fi

# Decrypt the backup file
gpg --batch $PASSPHRASE_OPTION -o "$SOURCE_DIR/$DECRYPTED_BACKUP_FILE" -d
"$SOURCE_DIR/$ENCRYPTED_BACKUP_FILE"
if [ $? -ne 0 ]; then
    echo -e "${RED}Fehler: Entschlüsselung der Datei $SOURCE_DIR/$ENCRYPTED_BACKUP_FILE
fehlgeschlagen.${NC}"
    exit 1
fi

# Extract the decrypted tar.gz file
tar -xzf "$SOURCE_DIR/$DECRYPTED_BACKUP_FILE" -C "$DEST_DIR"
if [ $? -ne 0 ]; then
    echo -e "${RED}Fehler: Entpacken der Datei $SOURCE_DIR/$DECRYPTED_BACKUP_FILE
fehlgeschlagen.${NC}"
    exit 1
fi

# Optionally, remove the decrypted tar.gz file after extraction
rm "$SOURCE_DIR/$DECRYPTED_BACKUP_FILE"

echo -e "${GREEN}Entschlüsselung und Entpacken abgeschlossen. Dateien befinden sich in
$DEST_DIR.${NC}"
```