# Linux Stuff

- Vaultwarden Tools
  - Vaultwarden Backup / Restore Scripts
- Docker Compose YAMLs
  - vaultwarden
  - nginx proxy manager
- OpenSSL
  - OpenSSL Nice2Know
- Graylog Stuff
  - Graylog <> MongoDB | X.509 Zertifikat erneuern
  - Graylog / OpenSearch OpenSSL Conf Dateivorlagen
- MariaDB
  - Slave Backup Wiederherstellung
  - MariaDB GTID Werte [xxx,xxx} aktivieren

# Vaultwarden Tools

# 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}"
```

# Docker Compose YAMLs

Docker Compose YAMLs

# vaultwarden

docker-compose.yml

```
version: '3.8'

services:
  vaultwarden:
    image: vaultwarden/server:latest
    container_name: vaultwarden
    restart: always
    ports:
          - 8081:80
    environment:
      - ADMIN_TOKEN=xyz
      - DOMAIN=https://vaultwarden.domain.de
      - WEBSOCKET_ENABLED=true
      - ORG_EVENTS_ENABLED=true
      - LOG_FILE=IP:514
      - LOG_LEVEL=error
      - EXTENDED_LOGGING=true
      - USE_SYSLOG=true
      - PUSH_ENABLED=true
      - PUSH_INSTALLATION_ID=
      - PUSH_INSTALLATION_KEY=
      - TIME_ZONE=Europe/Berlin
      - ROCKET_FORWARDING=true
      - ROCKET_TRUSTED_PROXIES=127.0.0.1,::1,172.23.0.0/16 # Anpassen an Ihr Docker-Netzwerk
    volumes:
      - /vaultwarden-folder/:/data
    logging:
      driver: syslog
      options:
        syslog-address: "udp://IP:514"
```

```
        tag: "Vaultwarden"
```

# nginx proxy manager

docker-compose.yml

```
version: '3.8'

services:



  nginx-proxy-manager:
    image: jc21/nginx-proxy-manager:latest
    container_name: nginx_proxy_manager
    restart: unless-stopped
    ports:
      - '80:80'  # Public HTTP Port
      - '443:443' # Public HTTPS Port
      - '81:81'  # Admin Web Port
    environment:
      DB_MYSQL_HOST: "db1"
      DB_MYSQL_PORT: 3306
      DB_MYSQL_USER: "USER"
      DB_MYSQL_PASSWORD: "PASSWORD"
      DB_MYSQL_NAME: "npm"
    volumes:
      - /npm-folder/data:/data
      - /npm-folder/letsencrypt:/etc/letsencrypt
    depends_on:
      - db



  db:
    image: jc21/mariadb-aria:latest
    container_name: npm_db
    restart: unless-stopped
    ports:
```

```
    - '3306:3306'
  environment:
    MYSQL_ROOT_PASSWORD: 'PASSWORD'  # Setzen Sie ein sicheres Passwort
    MYSQL_DATABASE: 'npm'
    MYSQL_USER: 'USER'
    MYSQL_PASSWORD: 'PASSWORD'
  volumes:
    - /npm-folder/mysql:/var/lib/mysql
```

> Freigabe von HTTP & HTTPS auf Fritzbox erstellen, damit der Reverse Proxy funktioniert

# OpenSSL

# OpenSSL Nice2Know

CSR Request aus CNF Datei erstellen:

```
openssl req -new -key admin.pem -out admin.csr -config openssl.cnf
```

-

aus CSR Datei ein Zertifikat (X.509 BASE64) von Windows PKI anfordern

```
certreq -attrib "CertificateTemplate:CertServer/Client" -submit vml-01.req
```

-

Überprüfung, ob Zertifikat auf Server gültig ist (in Verbindung mit RootCA)

```
openssl verify -CAfile root-ca.pem cert.pem
```

-

Zertifikatsinformationen anzeigen

```
openssl x509 -in /etc/ssl/cert.pem -noout -text
```

-

# Graylog Stuff

Graylog Stuff

# Graylog <> MongoDB | X.509 Zertifikat erneuern

**Dieses How To bezieht sich auf Graylog Server, die sich per Zertifikat an einer (oder mehreren) MongoDB (im TLS / SSL Modus) authentifizieren!**

> Es wird der vorhandene private Key des "OPENSEARCH-ADMIN" sowie die RootCA benötigt (wenn PKI Infrastruktur vorliegt).

> Das Zertifikat wird hier per Windows PKI ausgestellt

1. **per OpenSSL eine CSR von vorigen Zertifikat erstellen**

**OPENSEARCH-ADMIN** openssl.cnf Vorlage (Anpassung erforderlich!):

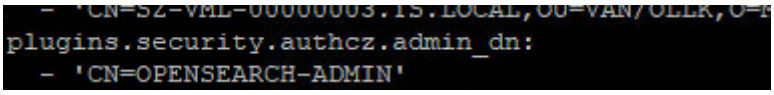
```
[ req ]
default_bits        = 2048
default_md          = sha256
prompt              = no
distinguished_name  = req_distinguished_name
req_extensions      = req_ext

[ req_distinguished_name ]
CN = OPENSEARCH-ADMIN

[ req_ext ]
extendedKeyUsage    = clientAuth, serverAuth
```

> Die CSR kann für die Zertifikatsanforderung nun genutzt werden

> Der DN muss mit dem vorigen DN übereinstimmen! Kann hier gefunden werden: /etc/opensearch/opensearch.conf

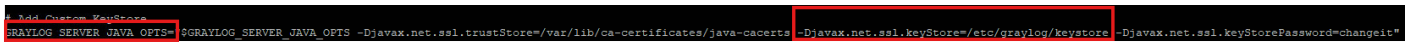



> openssl req -new -key SERVER1-privatekey.pem -out SERVER1-certRequest.csr -config openssl.cnf

> Das Zertifikat muss als BASE64 / X.509 vorliegen und darf nicht verschlüsselt sein!

**2. JAVA KEYSTORE von Graylog / Opensearch finden**

```
cat /etc/sysconfig/graylog-server
```




Das neue Zertifikat muss nun auf den Server kopiert werden (öffnen per **Texteditor / Notepad** und kopieren des Inhalts aus der frisch erstellten ***.cer**). Danach kann der Inhalt per ***vim / nano*** auf dem Graylog Server in die neue Datei **admin-cert.pem** eingefügt werden)

3. PKCS12 Datei erstellen, um diese in KEYSTORE hinzuzufügen

```
openssl pkcs12 -export \
    -out admin.p12 \
    -inkey admin-key.pem \
    -in admin-cert.pem \
    -certfile root-ca.pem  # Optional, wenn du ein CA-Bundle hast
```

> Das Kennwort der PKCS12 Datei muss gleich dem JAVA KEYSTORE sein!

4. p12 Datei in JAVA KEYSTORE importieren

```
sudo keytool -importkeystore \
    -srckeystore admin.p12 \
    -srcstoretype PKCS12 \
    -destkeystore /etc/graylog/keystore \
    -deststoretype JKS \
    -storepass changeit
```

>

> Server neustarten, danach ist das Zertifikat erneuert :)

# Graylog / OpenSearch OpenSSL Conf Dateivorlagen

**SERVER** openssl.cnf Vorlage (Anpassung erforderlich!)

```
# openssl-server-CSRcreator.cnf
[ req ]
default_bits        = 2048
default_md          = sha256
prompt              = no
distinguished_name  = req_distinguished_name
req_extensions          = req_ext
[ req_distinguished_name ]
C  = DE
ST = BRANDENBURG
L  = LUDWIGSFELDE
O  = UNTERNEHMEN XY
OU = IT ABTEILUNG
CN = SERVER1.DOMAIN.LOCAL
[ req_ext ]
extendedKeyUsage = clientAuth, serverAuth
subjectAltName = @alt_names
[ alt_names ]
DNS.1 = SERVER1.DOMAIN.LOCAL
IP.1 = 192.168.17.1
```

MariaDB

MariaDB

# Slave Backup Wiederherstellung

**Kurze Anleitung für die Wiederherstellung des Slaves, das per MariaBackup angefertigt wurde**

```
sudo systemctl stop mariadb@XY.service # Stoppen des Services
sudo rm -r /var/lib/mysql-XY-ABC # Löschen der Daten (Pfad kann variieren!)
sudo mariabackup --copy-back --target-dir=/tmp/27.08.2024/ --datadir=/var/lib/mysql-XY-ABC #
Wiederherstellung des DataDirs
sudo chown -R mysql:mysql /var/lib/mysql-XY-ABC && sudo systemctl start mariadb@XY.service #
Anpassung der Berechtigungen für MariaDB User sowie Start des Services

mysql -u sqladmin --port 1234 -h10.10.10.10 -p
```

> Anpassungen der Datenbank sind noch nötig, damit der Slave wieder funktioniert (siehe unten)

```
STOP SLAVE;
SET GLOBAL gtid_slave_pos = "XXX-XXX-XXX,XXX-XXX-XXXX"; #aus xtrabackup_binlog_info -> dritte
Reihe
CHANGE MASTER TO
MASTER_HOST="10.10.10.10",MASTER_PORT=3306,MASTER_USER="sql_repl",MASTER_PASSWORD="asdf1234",M
ASTER_USE_GTID=slave_pos;
START SLAVE;


show slave status\G;
```

MariaDB

# MariaDB GTID Werte [xxx,xxx} aktivieren

Mit diesen Befehlen lässt sich die GTID Werte für z.B. eine Replikation auslesen (xtrabackup_binlog_info).

```
STOP SLAVE;
CREATE SCHEMA IF NOT EXISTS bugfix;
START SLAVE;
show global variables like '%gtid%';
```

Dadurch kann die Replikation (Master / Slave) gestartet werden.