



# Step -1: Create Backup of encrypted img

cp "$IMAGE" "$BACKUP\_FOLDER"/"$TIMESTAMP".img

# Loop until file exists

while [ ! -e "$NUKE\_FILE" ]; do

if [ -e "$ALERT\_FILE" ]; then

echo "Minor event: check the logs:"

fi

sleep 2 # avoid busy-looping

done

echo "Major event: stopping and reverting:"

revert

revert() {

socker compose down

echo "Unmounting encrypted folder..."

sudo umount "$MOUNT\_POINT"

sudo cryptsetup close "$MAPPER\_NAME"

cp "$BACKUP\_FOLDER"/"$TIMESTAMP".img "$IMAGE"

exit 0

}

# Full supervision script:

* Alert / nuke suversision
  + “Alert” events create a special notice,
  + “Nuke” events cause an immediate shutdown and revert
* vectors database backup
  + backups are encrypted (via default luks aes-cbc-essiv:sha256, any other encryption supported by luks could be used)
* automatic revert in case of nuke

#!/bin/bash

set -e

# Paths

IMAGE="qdrant\_storage.img"

BACKUP\_FOLDER="/opt/qdrant\_backups/"

MAPPER\_NAME="qdrant\_storage\_enc"

MOUNT\_POINT="/mnt/qdrant\_storage"

ALERT\_FILE="/home/team-05/rag/app\_logs/alert\_file.txt"

NUKE\_FILE="/home/team-05/rag/app\_logs/nuke\_file.txt"

TIMESTAMP=$(date +%s)

cleanup() {

echo "Stopping Docker Compose..."

docker compose down

echo "Unmounting encrypted folder..."

sudo umount "$MOUNT\_POINT"

sudo cryptsetup close "$MAPPER\_NAME"

echo "Done. Storage is encrypted again."

exit 0

}

revert() {

socker compose down

echo "Unmounting encrypted folder..."

sudo umount "$MOUNT\_POINT"

sudo cryptsetup close "$MAPPER\_NAME"

cp "$BACKUP\_FOLDER"/"$TIMESTAMP".img "$IMAGE"

rm -f $NUKE\_FILE

exit 0

}

# Step -1: Create Backup of encrypted img

cp "$IMAGE" "$BACKUP\_FOLDER"/"$TIMESTAMP".img

# Step 0: check for malware/virus / skip keypress

# commented out for testing

sudo rkhunter --check --sk

# --- Step 1: Decrypt ---

echo "Decrypting storage..."sleep

sudo cryptsetup open "$IMAGE" "$MAPPER\_NAME"

# --- Step 2: Mount ---

sudo mkdir -p "$MOUNT\_POINT"

sudo mount /dev/mapper/"$MAPPER\_NAME" "$MOUNT\_POINT"

# --- Step 3: Run Docker Compose ---

# Make sure your docker-compose.yml uses $MOUNT\_POINT as a volume

docker compose up -d

# Wait for user to stop the compose stack

echo "Docker Compose running. Press Ctrl+C to shutdown..."

trap cleanup INT

# Loop until file exists

while [ ! -e "$NUKE\_FILE" ]; do

if [ -e "$ALERT\_FILE" ]; then

echo "Minor event: check the logs:"

rm -f $ALERT\_FILE

fi

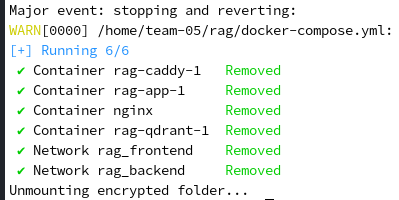
sleep 2 # avoid busy-looping

done

echo "Major event: stopping and reverting:"

revert

## Succesful “nuke” if nukefile exists:



## Extended Error logging:

@app.post("/upload")

async def upload(file: UploadFile = File(...)):

data = await file.read()

if not data:

with open(ERROR\_LOGFILENAME, "a") as f:

f.writelines("File with no data\n")

Path(ALERT\_FILE).touch()

## Regular Logging:

