# Rescue an encrypted volume

## Prepare DSM to get access through SSH with password login enabled

1. Prepare 2 files within top level directory `/volume2`. The name is optional but must be a directory outside of the encrypted `/volume1` and also outside of any operating system or Synology created directory so that these 2 files are kept intact when DSM get's updated, they must be treated as user files. In my setup Synology created `/volume2` automatically when I created the fully encrypted `/volume1`.
2. I wrote 2 shell files which run in `/bin/sh` or `/bin/bash`. They regularly check existence of `/volume1/homes`, and if the script detects the homes directory is not accessible it restarts the SSH daemon with password login enabled. To further safeguard it against misuse my DSM first contacts an external (cloud) server and requests the 2-factor-login key for my user account, which I activated with 2FA served through google authenticator. Without the 6 digit login key served from my cloud server, which serves it only if the request originates from my own fixed IP NAS, password login is not enabled.
3. There are 3 files required to make this concept work, the first 2 which are `enablepwd` and `queryotp` reside on my NAS under `/volume2/rescue/batch/`, the third one `synology.php` serves the OTP and resides on my cloud server.
4. Enable password login with `enablepwd`:

```
#!/bin/sh
#
# This script checks availability of a fully encrypted volume1 and
enables password authentication for SSH in case the volume is locked
# After restarting the SSH daemon it allows to login even though the
homes directory is not accessible
# It requires script 'queryotp' to provide a valid google authenticator
code without which it will do nothing
# Must be run as root, place it in the task scheduler with regular
execution
# (c) Bernard Condrau
# 2026-08-31: initial
#
VOLUME="/volume1/homes"
SSHD_CONFIG="/etc/ssh/sshd_config"
PWD_AUTH_STR="PasswordAuthentication yes"
BASE_PATH=$( cd -- "$( dirname -- "${BASH_SOURCE[0]}" )" &> /dev/null
&& pwd )
QUERY_OTP=$BASE_PATH"/queryotp"
LOG_FILE=$BASE_PATH"/../logs/enablepwd.log"
#
# check whether password authentication is enabled
head -n 1 $SSHD_CONFIG | grep -q "$PWD_AUTH_STR" && ENABLED="yes" ||
ENABLED=""
echo "$(date +'%Y-%m-%d %H.%M'): enablepwd task run" | tee -a $LOG_FILE
#
```

```
# check whether homes exists (e.g. volume1 is decrypted)
if [ -d "$VOLUME" ]; then
    echo "homes exists."
    if [[ -n "$ENABLED" ]]; then
        # disable password authentication
        $QUERY_OTP
        if [ $? -eq 0 ]; then
            sed -i "1d" $SSHD_CONFIG
            synosystemctl restart sshd
            echo "$(date +'%Y-%m-%d %H.%M'): \"$PWD_AUTH_STR\" removed"
| tee -a $LOG_FILE
            exit 0
        fi
    fi
else
    echo "homes does not exist."
    if [[ -z "$ENABLED" ]]; then
        # enable password authentication
        $QUERY_OTP
        if [ $? -eq 0 ]; then
            sed -i "1i $PWD_AUTH_STR" $SSHD_CONFIG
            synosystemctl restart sshd
            echo "$(date +'%Y-%m-%d %H.%M'): \"$PWD_AUTH_STR\" added" |
tee -a $LOG_FILE
            exit 0
        fi
    fi
fi
exit 1
```

5. Query OTP with `queryotp`:

```
#!/bin/sh
#
# Query OTP
#  must be run as root
# (c) Bernard Condrau
# 2026-08-31: initial
#
OTPQ="https://your.cloud.tld/html/synology.php?share=merkur-otp&auth=yo
ur-safe-authentication-password-which-should-have-24-or-more-
characters"
AUTH=0;
CODE="`wget -qO - $OTPQ`"
SYNO=$(sudo synootp username=bco code=$CODE)
if [[ $SYNO == 'auth ok' ]]; then
    echo 'authenticated'
    exit 0
else
    sleep 35
    CODE="`wget -qO - $OTPQ`"
```

```
    SYNO=$(sudo synootp username=bco code=$CODE)
    if [[ $SYNO == 'auth ok' ]]; then
        echo 'authenticated'
        exit 0
    fi
fi
echo 'not authenticated'
exit 1
```

6. Serving the OTP through wget to `synology.php`:

```
<?php
/*
 * php to deliver encryption key to remote NAS
 *
 * (c) 2026, Bernard Condrau
 *
 */

// Build an array of trusted IP addresses...
$trusted=array('nas.server1.ip.addr', 'nas.server2.ip.addr');

// check if request is made from valid ip address
if (in_array($_SERVER['REMOTE_ADDR'], $trusted) &&
isset($_SERVER['QUERY_STRING'])) {
    $password = array(
        'merkur-otp'    =>     ['your-safe-authentication-password-
which-should-have-24-or-more-characters',
                    shell_exec('/home/user/html/merkur-otp.sh') //
generate OTP
                    ],
    );
    // get the share name from the request
    $share = $_GET['share'];
    if (isset($password[$share]) && ($password[$share][0] ==
$_GET['auth'])) {
        echo $password[$share][1]; // echo the password, this will be
grabbed by wget
        exit(0);
    }
}
echo '';
?>
```

7. Compute the OTP in `merkur-otp.sh`:

```
#!/bin/bash
source /home/bco/html/pyotp-env/bin/activate
python3 /home/bco/html/pyotp-env/merkur-otp.py
deactivate
```

```
exit 0
```

8. Set up a task in *Task Scheduler* of your DSM.

# How to gain access to encrypted volumes when the KMIP Server was unavailable during DSM boot

- If you can re-establish connection to the KMIP Server you can do the following:
  1. SSH into DSM
  2. if you want to continue using the KMIP Server:

     ```
     reboot
     shutdown -r now
     ```

  3. if you do not want to continue using the KMIP Server:

     ```
     sudo /usr/syno/sbin/synoencvolume --auto-unlock-get-keys
     ```

  4. Log into DSM and open Storage Manager. You can now access DSM through it's web interface, but you cannot access your files.
  5. Reset the Encryption Key Vault in the *Global Settings* of the **Storage Manager** using the recovery key to disconnect your NAS from the KMIP Server.
  6. If your locked volume is not the same volume where your user home directories reside you can go to the **Storage Manager** and select the locked volume, then click the ellipsis (...) icon, and select *Unlock*. Upload your `.rkey` recovery file. The system will prompt you to enter a new vault password to recreate and repair the local key vault.

# Recover an encrypted volume using an SSH shell

- If you can not re-establish connection to the KMIP Server you can do the following:
  1. Identify the Target Device Node
     - Synology layers its storage using Linux mdadm (RAID) and Logical Volume Management (LVM). You need to find the logical volume path:

       ```
       sudo lvdisplay
       ```

     - Look for your volume path, e.g. `LV Path /dev/vg1/volume_1`
  2. Unlock the LUKS Container
     - Depending on what credentials you have, choose Option A or Option B
     - Option A: Using the Passphrase. Pass the LVM path to cryptsetup to open it. This will prompt you for the password:

       ```
       sudo cryptsetup luksOpen /dev/vg1/volume_1 mapped_volume
       ```

     - Option B: Using the Recovery Key (.rkey file). If you have the `.rkey` file saved on a USB drive or transferred via SSH to `/tmp/volume1.rkey`, use:

       ```
       sudo cryptsetup luksOpen /dev/vg1/volume_1 mapped_volume --
       key-file /tmp/volume1.rkey
       ```

- Note: mapped_volume is just a temporary virtual name you assign for the next step.

3. Mount the Filesystem
   - Synology volumes typically use the Btrfs or ext4 file systems. Create a temporary folder and mount the newly decrypted mapping:

```
sudo mkdir -p /mnt/recovery
mount /dev/mapper/mapped_volume /mnt/recovery
```

4. Access and Rescue Your Data
   - Your data is now fully unencrypted and accessible in plain text at `/mnt/recovery`. You can use command-line tools like `cp`, `rsync`, or `tar` to copy your files off to an external drive or another server.

# How to Regenerate and Download a Recovery Key

1. Open **Storage Manager** in Synology DSM
2. Go to **Storage** and click on your encrypted volume
3. Click the *Settings* button (or the three dots/action menu next to the volume)
4. Locate the **Recovery Key** section
5. Click *Regenerate*
6. Enter your Vault Password when prompted to authorize the action
7. The new recovery key file (`.rkey`) will download automatically to your computer's default download folder