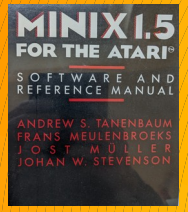
Recovering files from a ransomed NAS

Frans Meulenbroeks

Jan 18, 2024

About me


MINIX 1.5
FOR THE ATARI
SOFTWARE AND
REFERENCE MANUAL
ANDREW S. TANENBAUM
FRANS MEULENBROEKS
JOST MULLER
JOHAN W. STEVENSON

The problem

Hello.

This is DiskStation Security.

What happened?

- Your Network was not secure.
- Your Network-Attached Storage was compromised.

What does this mean? Where are my files?

- All your data has been encrypted and moved to a special volume.
- All your important documents have been downloaded.

What can I do to recover my data?

- If you want to recover your data, you have to send 0.01 Bitcoin to this wallet address:

14DpHGM5oBLrzbEU3yugYVXLcjxa3Jvi5y

Always double check the address when copy/pasting it !!!!!

- You have until the 15th of December 2023 to send the payment.
After this date the decryption will be almost impossible

The problem (2)

Can I still use my nas?

- Do not delete any files you find on your nas.
- Do not try to recover your data using any software as it will not work.
- Do not modify any volumes or storage pools on your nas.
- Do not write large amounts of data to your disk.
- Do not restart or power on/off your synology multiple times. It will result in archive corruption!

!!! Your ADMIN account was restricted for security purposes.
You can contact us to restore it !!!

You can think of this as a failed security audit.

We are professionals. This is a one time deal. We will decrypt a few files from your nas, as proof, if you need it.
We will send you the password immediately after the payment.
All Files, folders and subfolders will be unchanged.
We will even send you tips on how to strengthen your network security, to prevent any future attacks.

Thank you.

THE PROBLEM !!!

HE HAS NO BACKUP

The disk

- WD Red 8 TB
- 3.5” SATA 600
- 5400 RPM
- 128 MB Cache
- Seq write: 178 MB/s
- Seq read: 178 MB/s

  => time to seq read the whole disk: about 12.5 hrs.

Setup

Examining the disk: fdisk

Disk /dev/sdc: 7,28 TiB, 8001563222016 bytes, 15628053168 sectors
Disk model: WDC WD80EFZX-68U
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: A2003018-87B9-4748-B49B-1C7090566332

Device	Start	End	Sectors	Size	Type
/dev/sdc1	2048	4982527	4980480	2,4G	Linux RAID
/dev/sdc2	4982528	9176831	4194304	2G	Linux RAID
/dev/sdc5	9453280	15627846239	15618392960	7,3T	Linux RAID

Trying to mount

root@frans-desktop:/home/frans# mount /dev/sdc5 /mnt

mount: /mnt: unknown filesystem type 'linux_raid_member'.

root@frans-desktop:/home/frans# mdadm --assemble /dev/md2 /dev/sdc5 –force –raid-devices=2 –run

root@frans-desktop:/home/frans# mount /dev/md2 /mnt

mount: /mnt: unknown filesystem type 'LVM2_member'.

root@frans-desktop:/home/frans# mount /dev/dm-0 /mnt

mount: /mnt: wrong fs type, bad option, bad superblock on /dev/mapper/vg1000-lv, missing codepage or helper program, or other error.

root@frans-desktop:/home/frans# blkid /dev/dm-0

/dev/dm-0: LABEL="2018.09.28-19:44:48 v23739" UUID="70c9e467-9f82-494d-90d2-714e934b8e85" UUID_SUB="b5b3a94d-3c0a-480f-af39-e3816fd32613" TYPE="btrfs"

Check for errors

root@frans-desktop:/home/frans# fsck /dev/dm-0

fsck from util-linux 2.34

If you wish to check the consistency of a BTRFS filesystem or

repair a damaged filesystem, see btrfs(8) subcommand 'check'.

Check for errors (2)

```
$ btrfs check /dev/dm-0
Opening filesystem to check...
Checking filesystem on /dev/dm-0
UUID: 70c9e467-9f82-494d-90d2-714e934b8e85
[1/7] checking root items
[2/7] checking extents
Invalid key type(BLOCK_GROUP_ITEM) found in root(202)
ignoring invalid key   (repeat 856 times)
[3/7] checking free space cache
[4/7] checking fs roots
[5/7] checking only csums items (without verifying data)
[6/7] checking root refs
[7/7] checking quota groups skipped (not enabled on this FS)
found 2791355899904 bytes used, no error found
total csum bytes: 2722190328
total tree bytes: 3121020928
total fs tree bytes: 98222080
total extent tree bytes: 70713344
btree space waste bytes: 222662639
file data blocks allocated: 2812079058944
 referenced 2786350964736
```

Idea: use forensic/data recovery tools

- While searching bumped onto photorec
  https://www.cgsecurity.org/wiki/PhotoRec
- global working:

scan raw disk for signatures (similar to file(1))

For example, PhotoRec identifies a JPEG file when a block begins with:

0xff, 0xd8, 0xff, 0xe0

0xff, 0xd8, 0xff, 0xe1

or 0xff, 0xd8, 0xff, 0xfe

0xffd8 = jpeg start of image marker

exploit fact that data is written in chunks and that filesystems tend to write to adjacent sectors/blocks

Sometimes header contains file size; then a consistency check can be done, and the program knows how to write.

Give it a try

$ qphotorec

- After a minute or so said it recovered a jpg
  Just a thumbnail, but it is something.
- After a few minutes also recovered a full sized jpg (16Mpixel)

The next day….

But ….

The program stalled at 23.5%

QPhotoRec

PhotoRec 7.2-WIP, Data Recovery Utility, August 2023
Copyright (C) Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

Disk /dev/sdc - 8001 GB / 7452 GiB (RO) - WDC WD80EFZX-68U

Destination: /mnt

Pass 1 - Reading sector 3650617224/15618392960 23% 823268 files found

File family	Number of files recovered
jpg	608915
txt	136415
tx?	17441
tif	16457
mov	14751
png	6199
pdf	3638
apple	2768
pyc	2144
elf	2000
others	12540

Quit

Investigation

retried with very last version: still a problem

Open source so compiled with -g -gdb and ran under gdb

again stalled at the same place

interrupted fail at file_riff.c function check_riff_list in a seek.

Examine code

```
static void check_riff_list(file_recovery_t *fr, const unsigned int depth, const uint64_t start, const uint64_t end) {
  uint64_t file_size;
  if(depth>5)    return;
  for(file_size=start; file_size < end;)  {
    char buf[sizeof(riff_list_header_t)]; uint64_t next_fs;
    const riff_list_header_t *list_header=(const riff_list_header_t *)&buf;
    if(my_fseek(fr->handle, file_size, SEEK_SET)<0)  {      fr->offset_error=file_size; return;    }
    if (fread(buf, sizeof(buf), 1, fr->handle)!=1)    {      fr->offset_error=file_size; return;    }
    next_fs=file_size + (uint64_t)8 + le32(list_header->dwSize);
    if(next_fs > end)    {  fr->offset_error=file_size; return;   }
    if(memcmp(&list_header->dwList, "LIST", 4) == 0)  { check_riff_list(fr, depth+1, file_size + sizeof(riff_list_header_t),
next_fs-1); }
    file_size = next_fs;
    /* align to word boundary */
    file_size = (file_size&1);
  }}
```

Examine code

```
static void check_riff_list(file_recovery_t *fr, const unsigned int depth, const uint64_t start, const uint64_t end) {
  uint64_t file_size;
  if(depth>5)    return;
  for(file_size=start; file_size < end;)  {
    char buf[sizeof(riff_list_header_t)]; uint64_t next_fs;
    const riff_list_header_t *list_header=(const riff_list_header_t *)&buf;
    if(my_fseek(fr->handle, file_size, SEEK_SET)<0)  {      fr->offset_error=file_size; return;    }
    if (fread(buf, sizeof(buf), 1, fr->handle)!=1)    {      fr->offset_error=file_size; return;    }
    next_fs=file_size + (uint64_t)8 + le32(list_header->dwSize);
    if(next_fs > end)    {  fr->offset_error=file_size; return;   }
    if(memcmp(&list_header->dwList, "LIST", 4) == 0)  { check_riff_list(fr, depth+1, file_size + sizeof(riff_list_header_t),
next_fs-1); }
    file_size = next_fs;
    /* align to word boundary */
    file_size = (file_size&1);
  }}
```

Fix

Replace

file_size = (file_size&1);

with

file_size = ((file_size+1)&~1);

Rerun:
After 40 hours or so:

2.155.403 files recovered

8.471.497.056 sectors contain unknown data
22841 invalid files found and rejected.

Stats (2.152.577 files)

1162204 jpg
687772 txt
47838 xml
29850 mp4
27518 apple
26224 nef
16384 png
13917 sqlite
12600 pdf
11752 html
10740 mp3

Nikon raw image


GREAT SUCCESS!
quickmeme.com

Why did this work?

- Data on NAS were rarely deleted, so most files were written continuously
- Disk was untouched after hack
- Hacker did not scrub remaining disk

But why couldn’t I mount the disk?

- https://www.reddit.com/r/synology/comments/u6y5qm/has_anyone_found_a_solution_for_mounting_synology/
- Linux 5.4+ blocked mouting of btrfs filesystem with unsupported subvolume flags
- Synolog uses additional subvolume flags
- Fix from reddit: install 4.15 kernel
  apt install linux-image-4.15.0-107-generic
- And select this kernel during reboot
- Hooray, now I can mount the filesystem!

Examining the disk

- Disk contains folder _____ with 3 files in it:
  homes.7z photo.7z web.7z

$ 7z l homes.7z

2023-12-09 16:33:28 505120666087 505121366288 126458 files, 10902 folders

$ 7z l photo.7z

2023-11-20 16:34:23 2268082026235 2268083237456 163049 files, 2756 folders

$ 7z l web.7z

2023-08-05 09:41:47 4497848670 4497866800 2523 files, 109 folders

Total 2.6 TB

Definitely not all, I recovered 2.156.403 files, and the zips contain less than 300.000

Examining photo.7z

7z l photo.7z

Scanning the drive for archives:

```
1 file, 2268086284285 bytes (2113 GiB)

--

Path = photo.7z

Type = 7z

Physical Size = 2268086284285

Headers Size = 3046829

Method = Copy 7zAES

Solid = -

Blocks = 162595
```

Examining photo.7z (2)

```
   Date      Time    Attr         Size   Compressed  Name
------------------- ----- ------------ ------------  ------------------------
2020-10-15 23:07:25 DR...            0            0  006=Fotos van laptop hp
2021-05-03 19:23:38 DR...            0            0  006=Fotos van laptop hp/2011
[....]
2012-02-25 03:37:02 .R..A      2395235      2395248  006=Fotos van laptop hp/2011/IMG_0180.JPG
2012-03-10 04:26:20 .R..A      3104733      3104736  006=Fotos van laptop hp/2011/IMG_0280.JPG
```

Examining photo.7z (3)

7z l -slt photo.7z

```
Path = 006=Fotos van laptop hp
Size = 0
Packed Size = 0
Modified = 2020-10-15 23:07:25
Attributes = RD_ d---------
CRC =
Encrypted = -
Method =
Block =
```

Examining photo.7z (4)

7z l -slt photo.7z

```
Path = 006=Fotos van laptop hp/2011/IMG_0180.JPG
Size = 2395235
Packed Size = 2395248
Modified = 2012-02-25 03:37:02
Attributes = RA_ -r-xr-xr-x
CRC = 5FD5D7FE
Encrypted = +
Method = Copy 7zAES:19
Block = 0
```

Examining photo.7z (4)

7z l -slt photo.7z

```
Path = 006=Fotos van laptop hp/2011/IMG_0180.JPG
Size = 2395235
Packed Size = 2395248
Modified = 2012-02-25 03:37:02
Attributes = RA_ -r-xr-xr-x
CRC = 5FD5D7FE
Encrypted = +
Method = Copy 7zAES:19
Block = 0
```

Idea: use CRC to rename files

- Calculate crc + sha1 hash
  find . -type f -exec rhash -C -H {} \; > ~/nas/HASH
- Could have calculated sha1 on e.g. first 100k but needed the crc

```
recup_dir.1000/f2799110184.jpg  90697909  6966f5c762a144d038a35d86a70f5fe4bfb94955
recup_dir.1000/f2799140592.jpg  90697909  6966f5c762a144d038a35d86a70f5fe4bfb94955
recup_dir.966/f2788979544.jpg   90697909  6966f5c762a144d038a35d86a70f5fe4bfb94955
recup_dir.996/f2798244456.jpg   90697909  6966f5c762a144d038a35d86a70f5fe4bfb94955
recup_dir.1021/f2807547280.jpg  90697909  6966f5c762a144d038a35d86a70f5fe4bfb94955
About 1536847 unique files
recup_dir.535/t1840635512.jpg  0119ae8a  5ef7469bb10d430af533b7ad92d309d8b29e302e
recup_dir.446/f1544125864.mp3  0119ae8a  f86e42444588286089a9e31986a55ec0f4d957ed
614 crc's appear more than once.
```

Filter out duplicates

- Duplicate hashes:

```
recup_dir.1000/f2799110184.jpg 90697909 6966f5c762a144d038a35d86a70f5fe4bfb94955
recup_dir.1000/f2799140592.jpg 90697909 6966f5c762a144d038a35d86a70f5fe4bfb94955
```

ChatGPT to the rescue:
awk '!seen[$3]++' your_file.txt > new_file.txt

Solution:
mkdir DUP; sort HASH > Hs
awk '!seen[$3]++' Hs | sort | tee Hsus | comm -1 -3 - Hs |
while read a b; do mv $a DUP/$a; done

Filter out duplicate CRC’s

```
recup_dir.535/t1840635512.jpg  0119ae8a  5ef7469bb10d430af533b7ad92d309d8b29e302e
recup_dir.446/f1544125864.mp3  0119ae8a  f86e4244458828608 9a9e31986a55ec0f4d957ed
```

- ChatGPT did not give a good solution

```
mkdir DUPCRC
awk '{cnt[$2]++; val[$2]=$0} END {for (item in cnt)
if (cnt[item] == 1) print val[item]}' Hsus | sort | tee Hsuc |
comm -13 – Hsus |
while read a b;  do mv $a DUPCRC/$a; done
```

Final steps

- Extract dirs from 7z l -slt output and create these.
- extract crc + filename from 7z l -slt output
- Join hash file and processed 7z output
- Filter out unneeded columns and generate command to link photorec filename to 7z filename

Examining photo.7z (3)

7z l -slt photo.7z

```
Path = 006=Fotos van laptop hp
Size = 0
Packed Size = 0
Modified = 2020-10-15 23:07:25
Attributes = RD_ d---------
CRC =
Encrypted = -
Method =
Block =
```

Examining photo.7z (4)

7z l -slt photo.7z

```
Path = 006=Fotos van laptop hp/2011/IMG_0180.JPG
Size = 2395235
Packed Size = 2395248
Modified = 2012-02-25 03:37:02
Attributes = RA_ -r-xr-xr-x
CRC = 5FD5D7FE
Encrypted = +
Method = Copy 7zAES:19
Block = 0
```

Split 7z l -slt output

```
> files
> dirs
tail -n +15 photo_7zslt |  grep -e "Path" -e "CRC" |
while read path
do
  read crc
  if [ "$crc" = "CRC =" ]; then
    echo $path >> dirs
  else
    echo $crc $path >> files
  fi
done
```

Creating dirs

dirs file now looks like:
Path = 006=Fotos van laptop hp
Path = 006=Fotos van laptop hp/2011
…
Creating the dirs:

```
cat dirs | while read pathstring eqsym dir
do
   mkdir \”$dir\”
done
```

Linking files

files looks like:

```
CRC = 5FD5D7FE Path = 006=Fotos van laptop hp/2011/IMG_0180.JPG
CRC = FCCA2052 Path = 006=Fotos van laptop hp/2011/IMG_0280.JPG
```

Hsuc looks like:

```
recup_dir.1000/f2799110448.jpg  b763c738  b15ba8269b6f30b67f6a9751be089ea58ec74ae2
recup_dir.1000/f2799111400.jpg  a1e05e7d  a79e4bbdc14837277b951fea501daa50d9f940f0
```

sort files > filess; sort -k 2 Hsuc > Hsucs
join -i -1 2 -2 3 Hsucs filess > joinfile

joinfile looks like :

```
0000e04f recup_dir.534/f1839002240.jpg 23ebad3442c00ff23cbe9e12e1178f8faaf23fca \
CRC = Path = Vakantie 2010/IMG_3971.JPG
```

cat joinfile | while read crc src hash crctxt eqsym pathtxt eq2sym dest
do
  ln “$src” “$dst”
done


"That's all Folks!"