# Forensic Focus



**compressed files**

Home Shrinking the gap: carving NTFS *compressed files*

# Shrinking the gap: carving NTFS-compressed files

Page: 1/4

Recovering deleted NTFS-compressed files

By Joachim Metz
Hoffmann Investigations
www.hoffmannbv.nl

1.0 Joachim Metz September 2, 2009 Initial version.

**Summary**

An important part of digital forensic investigation is the recovery of data, particulary files. The recovery of data and files highly depends on the recovery tooling used. This paper focusses on a part of file carving where most of the currently available data and file recovery tools do not provide support for, namely recovering NTFS-compressed data.

The paper also provides an overview of the consequences of NTFS compression on data and investigative techniques. In this paper a basic understanding of NTFS is assumed. For more information about NTFS check the references.

**Background**

In 2006 the Digital Forensic Workshop (DFRWS) provided a challenge in which fragmented files were recovered from a randomized data image. Joachim Metz and Robert-Jan Mora developed a proof of concept tool called ReviveIt (revit, which was later renamed to revit06) [DFRWS06]. Revit uses a carving technique called Smart Carving, which is more of a conceptual technique that uses the combination of other techniques such as: file structure based carving.

The first implementation (currently known) was by Nick Mikus in foremost [MIKUS05]. This allowed foremost far better results than before when using header/footer and header/maximum (file) size carving techniques.

Revit06 provided interesting results. Far better than carving tools that used the header/footer and header/maximum (file) size carving methods like at that time. Note that today carvers like scalpel and EnCase still use this naive carving approach. This called for some research into the quality of file carvers. In late 2006 Bas Kloet was working on the subject of measuring and improving the quality of carving tools. His findings were published in his Master thesis 'Measuring and Improving the Quality of File Carving Methods' [KLOET07].

Early 2007 the DFRWS came with a new and improved file carving challenge [DFRWS2007]. This motivated Joachim Metz and Robert-Jan Mora to start working on revit07 in combination with Bas Kloet, while doing his research on the quality of file carvers. This version of revit was also sent in for the 2007 challenge.

In the meantime a lot of work has been done to improve carving tools, e.g. PhotoRec by Christophe Grennier is a very useful and qualitative carver with support for a vast amount of files and file systems.

However, in a recent investigation we needed to recover NTFS-compressed data and little of the currently available carving tools provided support for this. This paper discusses a technique to recover NTFS-compressed and its application in forensic carving tools.

**Table of Contents**



## 1. NTFS compression

Before discussing the details of recovering NTFS-compressed data, it is necessary to provide an overview of how NTFS compression works and what its effects are on binary data.

In the digital forensics field not much has been published about NTFS compression.

[SANDERSON02] provides a discussion of NTFS compression and digital forensic investigation, but does not go into

details of the compression algorithm used. Also [CARRIER05] does not provide any details of the NTFS compression algorithm.

Luckily for us, the Linux NTFS project has provided for elaborate documentation about NTFS including its compression algorithm [RUSSON05].

### 1.1. Block-based storage

It is important to know is that NTFS uses cluster blocks (in other documentation the cluster block is sometimes referred to as just cluster) to store file data. These cluster blocks are commonly 4096 bytes of size. Information about the usage of these cluster blocks is maintained in the allocation bitmap and in the Master File Table (MFT), in particular the data runs. NTFS-compressed files can consist of both compressed, uncompressed and sparse cluster blocks. NTFS uses data runs to determine which cluster blocks make up the file data. These data runs also indicate which cluster blocks are compressed, uncompressed or sparse. Without these data runs there is no other information that can provide us with this information.

### 1.2. Block-based compression

The block-based storage technique is also used within NTFS compression. Data is compressed at 16 cluster blocks at a time.


*Illustration 1: Successful compression of 16 cluster blocks*

If 16 cluster blocks of the original data (above in blue) can be compressed to 11.5 cluster blocks (above in red) the data is stored in 12 cluster blocks. Note that the last block contains slack space (above in yellow), which will be referred to as NTFS compression slack space. The 16 compressed cluster blocks are also referred to as a compression unit.


*Illustration 2: Unsuccessful compression of 16 cluster blocks*

If 16 cluster blocks of the original data (above in blue) cannot be compressed to fewer than 16 cluster blocks (above in red) the data is stored as the original 16 uncompressed blocks.

If the 32 cluster blocks would be part of a single file, the file would be stored similar to the diagram below.

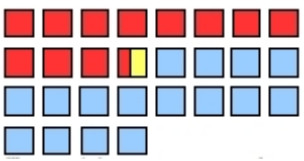

*Illustration 3: 2 compression units stored*

[SANDERSON02] states that files, that were initially stored uncompressed and then compressed, leave previously

allocated cluster blocks in place:

> On the face of it, it would seem logical for the compressed data from the second compression unit to be saved to disk immediately after the first compression unit. It is my belief that Microsoft has chosen this method for reasons of efficiency. By leaving the saved clusters within a compression unit initially empty means that if data is added to the first compression run then it just needs to be expanded into the free space that was left after compression.

This would mean that the compressed file would contain 4.5 cluster blocks of NTFS compression slack space.

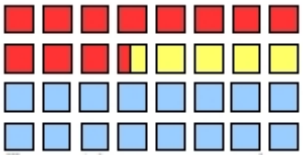

*Illustration 4: 2 compression units stored*

In our initial review of carving NTFS-compressed files we mainly found compressed units with less than 1 block NTFS compression slack space (such as in illustration 3). This could be dependent on how the compressed file was created or by the operating system and/or file system version used. Our tests were done on an forensic copy of a Microsoft Windows XP SP2 system containing NTFS version 5.1. It is unknown which operating system and version of NTFS were used in [SANDERSON02], probably Microsoft Windows 2000 with NTFS version 5.0. However more research is needed to get more conclusive answers about the allocation behavior of NTFS compressed files.

Next Page (2/4) ⇨