Forensics

Setup

First I created a virtual machine running Windows XP Professional 32-bit with a small 4 GB hard drive attached. This small hard drive was separate from the system hard drive and so contained only files used for testing. It was filled to just above 3 GB with a mixture of files containing ordinary text, source code, images, movies and executable files. The XP VM was then shut down.

Next an Ubuntu Server 12.04 64-bit virtual machine was installed on the same server. Because of my plan for how to structure a variety of tests, I chose to set up an iSCSI target on a VM with Solaris 11 so as to employ the deduplication feature of the ZFS file system.

```
sudo apt-get install sleuthkit
sudo apt-get install scalpel
sudo apt-get install dcfldd
wget http://log2timeline.net/files/log2timeline_0.63.tgz
tar -xzvf log2timeline_0.63.tgz
```

I had to turn off the Ubuntu server VM because the hard drive from the XP VM was in IDE configuration and the Ubuntu VM didn't have that kind of controller attached. With the new drive in place I noticed a problem: the XP drive was labeled as SFS according to fdisk. I must have designated the drive as "dynamic" but Linux doesn't care one way or the other and just mounts it as NTFS. Good job!

Next I made a list of all files on the drive in question:

```
cjp@forensics:~$ less pristine_XP1_2012-06-13_00\:57.txt
```

ZFS with deduplication didn't work very well. Might be because the ZFS host was a very low-powered VM.

Information gathering

Deleted several sets of data.

By diffing the ls -lhR output from the pristine hard drive image and the image with deleted files it was demonstrated that the deletes had been performed as far as the NTFS file system was concerned.

```
cjp@forensics:~$ sudo mmls /dev/sda
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

     Slot    Start        End          Length       Description
00:  Meta    0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----   0000000000   0000000062   0000000063   Unallocated
02:  00:00   0000000063   0008385929   0008385867   Win LVM / Secure FS
```

```
(0x42)
03:  -----   0008385930   0008388607   0000002678   Unallocated

cjp@forensics:~$ fls -o 63 ImageStore/deletes0613.bin
r/r 4-128-4:    $AttrDef
r/r 8-128-2:    $BadClus
r/r 8-128-1:    $BadClus:$Bad
r/r 6-128-1:    $Bitmap
r/r 7-128-1:    $Boot
d/d 11-144-4:   $Extend
r/r 2-128-1:    $LogFile
r/r 0-128-1:    $MFT
r/r 1-128-1:    $MFTMirr
r/r 9-128-8:    $Secure:$SDS
r/r 9-144-11:   $Secure:$SDH
r/r 9-144-14:   $Secure:$SII
r/r 10-128-1:   $UpCase
r/r 3-128-3:    $Volume
d/d 728-144-6:  Documents
d/d 42-144-1:   RECYCLER
d/d 29-144-7:   Software
d/d 27-144-6:   System Volume Information
-/d * 14471-144-1:      Media
d/d 15744:      $OrphanFiles
```

This output has the form

(file type according to file structure)/file type according to metadata)

So r/r means it's a regular file and both the file and the metadata reflects this. The next item is the metadata address and can consist of one, two or three values. One value means we just get an address. Two values means we get an address and a type. The third value - if it exists - constitutes an attribute ID.

To use fls to list all deleted files:

```
cjp@forensics:~$ fls -o 63 ImageStore/deletes0613.bin -rd
* Lots and lots of files *
...
r/r * 6106-128-1:       Documents/C/RobustLoader/Makefile
r/r * 6109-128-4:       Documents/C/RobustLoader/Makefile~
r/r * 6110-128-3:       Documents/C/RobustLoader/robust
-/d * 14471-144-1:      Media
-/r * 14472-128-4:      Media/MST3K - 0702 - 20010604 - The Brute Man.avi
```

The following invocation writes the same data with time-stamps and a prepended path to a file under delete_log/deleted_files1.txt

```
cjp@forensics:~$ fls -o 63 ImageStore/deletes0613.bin -rd -m E:\\ >>
delete_log/deleted_files1.txt
```

This can then be fed to mactime:

```
cjp@forensics:~$ mactime -b delete_log/deleted_files1.txt -d >>
delete_log/timeline1.csv
```

which in turn generates a comma-separated value file that can be used in a spreadsheet program. It does however not handle fls output in the long format well, discarding information about access and modification times. Instead a custom awk-script might be good for turning timestamps into human-readable dates.

To my surprise gawk is not awk! I thought gawk was simply (GNU) awk but it's a separate install in Ubuntu. Anyway, I swiped some code from http://unstableme.blogspot.se/2011/06/awk-convert-epoch-to-date-in-same-file.html and made a gawk script:

```
BEGIN {FS=OFS="|"}{
for( i = 8; i<=11; i++ )
{
        $i=strftime("%F %R",$i)
}
}
{print}
```

Then you execute the script like so:

```
cjp@forensics:~$ gawk -f mod_time_conv.awk delete_log/deleted_files2.txt
0|E:\\/Documents/C/RobustLoader/robust (deleted)|6110-128-3|r/rrwxrwxrwx|0|
0|9656|2012-06-13 01:12|2011-02-13 19:20|2012-06-12 20:30|2012-06-12 20:30
```

Here the dates are(from left to right): mod_time, acc_time, chg_time, cre_time. Not that when you delete a folder containing a number of files, the files won't have a chg_time corresponding to the time of deleting the folder. The folder however will, so you will have to deduce that a deleted file may have become deleted at a later date than the file metadata itself says.

Finally some loose information gathering tools:

```
/usr/bin/img_cat
/usr/bin/img_stat
/usr/bin/mmls
/usr/bin/mmstat
/usr/bin/mmcat
/usr/bin/blkcalc
/usr/bin/blkcat
/usr/bin/blkls
/usr/bin/blkstat
/usr/bin/ffind
/usr/bin/fls
/usr/bin/fsstat
/usr/bin/ifind
/usr/bin/istat
```

```
/usr/bin/jcat
/usr/bin/jls
/usr/bin/hfind
/usr/bin/srch_strings
/usr/bin/sigfind
/usr/bin/sorter
/usr/bin/tsk_recover
/usr/bin/tsk_loaddb
/usr/bin/tsk_comparedir
/usr/bin/tsk_gettimes
/usr/bin/icat-sleuthkit
/usr/bin/ils-sleuthkit
/usr/bin/mactime-sleuthkit

cjp@forensics:~$ fsstat -o 63 ImageStore/deletes0613.bin
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: NTFS
Volume Serial Number: 0894F50994F4F9D0
OEM Name: NTFS
Volume Name: TestStore
Version: Windows XP

METADATA INFORMATION
--------------------------------------------
First Cluster of MFT: 262144
First Cluster of MFT Mirror: 524116
Size of MFT Entries: 1024 bytes
Size of Index Records: 4096 bytes
Range: 0 - 15744
Root Directory: 5
CONTENT INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 4096
Total Cluster Range: 0 - 1048232
Total Sector Range: 0 - 8385865

$AttrDef Attribute Values:
$STANDARD_INFORMATION (16)   Size: 48-72   Flags: Resident
$ATTRIBUTE_LIST (32)   Size: No Limit   Flags: Non-resident
$FILE_NAME (48)   Size: 68-578   Flags: Resident,Index
$OBJECT_ID (64)   Size: 0-256   Flags: Resident
$SECURITY_DESCRIPTOR (80)   Size: No Limit   Flags: Non-resident
$VOLUME_NAME (96)   Size: 2-256   Flags: Resident
$VOLUME_INFORMATION (112)   Size: 12-12   Flags: Resident
$DATA (128)   Size: No Limit   Flags:
$INDEX_ROOT (144)   Size: No Limit   Flags: Resident
$INDEX_ALLOCATION (160)   Size: No Limit   Flags: Non-resident
$BITMAP (176)   Size: No Limit   Flags: Non-resident
$REPARSE_POINT (192)   Size: 0-16384   Flags: Non-resident
$EA_INFORMATION (208)   Size: 8-8   Flags: Resident
$EA (224)   Size: 0-65536   Flags:
$LOGGED_UTILITY_STREAM (256)   Size: 0-65536   Flags: Non-resident
```

File recovery

Scalpel was interesting but didn't cooperate with the NTFS metadata. It just extracted chunks of data it thought were jpeg, docs etc and gave it a number with the corresponding file suffix.

The autopsy program turns out to have a web interface so one doesn't need to install a graphical desktop environment. It is launched like this:

```
cjp@forensics:~$ autopsy -d /home/cjp/cases/ 192.168.0.53

================================================================================


                        Autopsy Forensic Browser
                    http://www.sleuthkit.org/autopsy/
                               ver 2.24

================================================================================
Live Analysis Mode
Start Time: Wed Jun 13 15:06:26 2012
Remote Host: 192.168.0.53
Local Port: 9999
```

Open an HTML browser on the remote host and paste this URL in it:

```
http://forensics:9999/38705440292992625455/autopsy
```

Keep this process running and use <ctrl-c> to exit

Then you navigate to the address given and choose a disk-image to start examining. It will list deleted files and let you download them. I haven't found a bulk recovering option for deleted files in autopsy yet.

The ntfsundelete program however gave me hope for such a process. First we need kpartx:

```
cjp@forensics:~$ sudo apt-get install kpartx
[ install output ]
cjp@forensics:~$ sudo kpartx -av ImageStore/deletes0613.bin
add map loop0p1 (252:2): 0 8385867 linear /dev/loop0 63
cjp@forensics:~$ sudo ntfsundelete --scan /dev/mapper/loop0p1
Inode    Flags  %age  Date          Size  Filename
------------------------------------------------------------
ntfs_mst_post_read_fixup_warn: magic: 0x00000000  size: 1024   usa_ofs: 0
usa_count: 65535: Invalid argument
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
16       F..!     0%  1970-01-01         0  <none>
ntfs_mst_post_read_fixup_warn: magic: 0x00000000  size: 1024   usa_ofs: 0
```

```
usa_count: 65535: Invalid argument
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
17       F..!     0%  1970-01-01         0  <none>
ntfs_mst_post_read_fixup_warn: magic: 0x00000000  size: 1024   usa_ofs: 0
usa_count: 65535: Invalid argument
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
18       F..!     0%  1970-01-01         0  <none>
ntfs_mst_post_read_fixup_warn: magic: 0x00000000  size: 1024   usa_ofs: 0
usa_count: 65535: Invalid argument
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
19       F..!     0%  1970-01-01         0  <none>
ntfs_mst_post_read_fixup_warn: magic: 0x00000000  size: 1024   usa_ofs: 0
usa_count: 65535: Invalid argument
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
20       F..!     0%  1970-01-01         0  <none>
ntfs_mst_post_read_fixup_warn: magic: 0x00000000  size: 1024   usa_ofs: 0
usa_count: 65535: Invalid argument
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
21       F..!     0%  1970-01-01         0  <none>
ntfs_mst_post_read_fixup_warn: magic: 0x00000000  size: 1024   usa_ofs: 0
usa_count: 65535: Invalid argument
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
22       F..!     0%  1970-01-01         0  <none>
ntfs_mst_post_read_fixup_warn: magic: 0x00000000  size: 1024   usa_ofs: 0
```

```
usa_count: 65535: Invalid argument
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
ntfs_attr_find: Corrupt inode (-1): Input/output error
23       F..!     0%  1970-01-01         0  <none>
5733     D...     0%  2012-06-13         0  Notes
5736     FR..   100%  2009-07-23       273  LAN_prep.txt
[ lots and lots of files ]
5737     FN..   100%  2012-05-25       747  Network.txt
6285     FR..   100%  2011-01-31       336  Placeholder.cpp
14461    FN..   100%  2010-12-16     21577  meds4.ods
14462    FN..   100%  2010-06-30     20872  meds3.ods
14465    FN..   100%  2010-02-03     16654  meds.ods
14471    D...     0%  2012-06-13         0  Media
14472    FN..   100%  2011-04-23 720388096  MST3K - 0702 - 20010604 - The
Brute Man.avi
15654    FN..   100%  2004-08-15     84556  PICT1269.JPG
15655    FN..   100%  2004-08-15     61664  PICT1268.JPG
15665    FN..   100%  2004-08-15     51034  PICT1267.JPG
ntfs_mst_post_read_fixup_warn: magic: 0x44414142  size: 1024   usa_ofs: 0
usa_count: 65535: Invalid argument
15743    F..!     0%  1970-01-01         0  <none>
```

Files with potentially recoverable content: 404

```
cjp@forensics:~$ sudo ntfsundelete -Pv --undelete /dev/mapper/loop0p1
--destination ntfs_files/ --match "*.txt"
```

Running the find-command again with the checksumming exec-part it was clear that the files did not exactly correspond to the originals. Upon inspection it was clear that this was partially due to the padding of files to nearest block limit. Using the --optimistic flag wasn't good but it turns out the flag I was thinking of was --truncate, which works perfectly!