Testing Forensic Hash Tools on Sparse Files

IMF 2007, Stuttgart, September 2007

Harish Daiya
IIT Kharagpur, India

Maximillian Dornseif
Hudora GmbH, Germany

Felix C. Freiling
University of Mannheim, Germany

Digital Evidence vs. Physical Evidence

- **Digital evidence** is evidence which is based on data stored or transmitted using a computer system [Casey, p. 12]
- The primary manifestation of digital evidence is **physical evidence**
  - Magnetisation on surface of hard disk
  - Electronic signals on data or bus cable
  - State of transistors of main memory
- Digital evidence must be extracted and translated into something readable before it can be used

Digital Evidence and Abstraction

- **Tools** are needed to extract digital evidence from physical evidence
- Tools only present an **abstraction** of physical evidence
- **Several levels** of abstraction are standard in modern systems
- Each level introduces a **new interpretation** of data

Analysis on Different Layers: Hard Disks

- Digital investigations can be performed at **different levels of abstraction**
- Main levels for **hard disks** [Carrier, p. 10ff]:
  - **Physical storage** medium level (raw hard disk sectors)
  - **Volume level** (collection of sectors accessible to an application)
  - **File system level** (collection of data structures allowing an application to read and write files)
- Often, the **same tools** are used on these different levels

Example: Forensic Hash Tools

- Input: stream of bits
- Output: cryptographic hash
  - Hash value is an unforgeable "fingerprint" of the original bit stream
  - Used to protect integrity of evidence

- Since input is just a stream of bits, forensic hash tools can be used on **all three analysis layers** for hard disks
  - Physical
  - Volume
  - File System

Problems

- Tools may give **different results** when applied to the **same evidence** at **different levels of abstraction**

- Example: Two segments of a hard disk containing two files A and B
  - Hash tool at file system level says: A and B are **identical**
  - Hash tool at volume level says: A and B are **not identical**

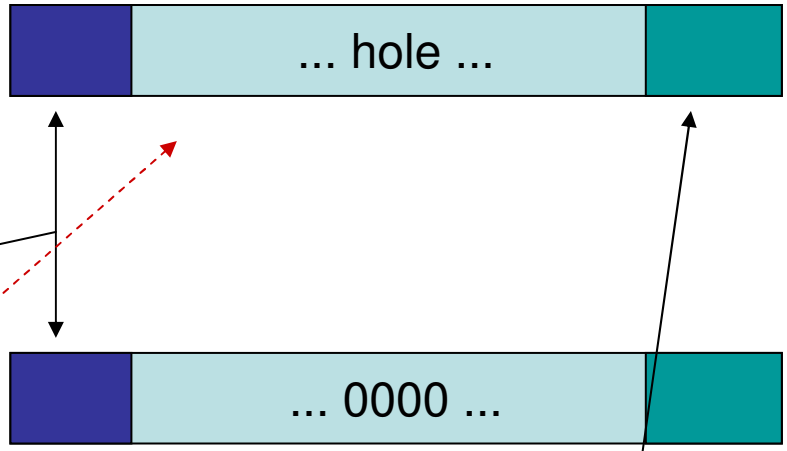
- Unconscious investigators may come to different conclusions

Outline

- Motivation
- Problems of Abstraction and Interpretation with Digital Evidence
- **Example: Hash tools applied to sparse files**
- Discussion

Sparse Files

```
...

#define beg_str   "beg_string"
#define end_str   "end_string"
#define withhole  "withhole"
#define withzero  "withzero"
#define n         100000        /* size of hole */

int main() {
  int offset=n, fd1, fd2, i;
  char *buff1 = beg_str;
  char buff2[offset];
  char *buff3 = end_str;

  fd1=creat(withhole, 0);
  fd2=creat(withzero, 0);

  /* create common header */
  write(fd1,buff1,10);
  write(fd2,buff1,10);

  for(i=0;i<offset;i++)
    buff2[i]=0;
  lseek(fd1,offset,SEEK_CUR);
  write(fd2,buff2,offset);

  /* create common trailer */
  write(fd1,buff3,10);
  write(fd2,buff3,10);
  return(0);
}
```



Handling of Sparse Files

- Depends on the file system
- At file system level
  - Metadata (e.g. block pointers) indicates that file is sparse
  - `read()` returns sequence of zeros

- Files are **different** at the **volume level**
- Files are **the same** at the **file system level**

File Systems and Sparse Files

- File systems that support sparse files:
  - Ext2/ext3
  - Reiserfs
  - JFS
  - NTFS
  - ...
- File systems that do not support sparse files:
  - FAT
  - Minix
  - ...

Can Sparse Files be Detected?

- Unix command `du` (disk usage)
  - `du file`
    Outputs number of kilobytes which `file` consumes at the volume level
  - `du --apparent-size file`
    Outputs number of kilobytes which `file` "seems" to consume
    - Usually smaller due to internal fragmentation
    - May be larger due to sparse files, indirect blocks, etc.

Experiment

- We created simple **file system images** for different types of file systems
- In each image we created **two files** (a sparse file and a non-sparse file)
- We ran several different **hash tools** on both files and compared the output
- We invoked `du` and `du --apparent-size` on both files

- **Example** on how to create and mount images in Linux (here ext3 file system mounted as loopback device):

```
# dd if=/dev/zero of=image bs=1M count=10
# mkfs.ext3 image
# mount -o loop image /mnt
```

Results (1/2)

- Files have **different sizes on volume** level for most file systems

filesystem	img size	du		du –app size	
		withhole	withzero	withhole	withzero
ext3	16MB	3KB	99KB	98KB	98KB
reiserfs	76MB	8KB	100KB	98KB	98KB
vfat	11MB	98KB	98KB	98KB	98KB
jfs	17MB	8KB	100KB	98KB	98KB
minix	5.1MB	99KB	99KB	98KB	98KB
ext2	11MB	3KB	100KB	98KB	98KB
msdos	5.1MB	98KB	98KB	98KB	98KB

Results (2/2)

- **Hashes** of sparse and non-sparse file were **the same** for all tools
  - Hash tools invoked on file system level

hashtool	hash
md5sum	458b5ebc8c1bf7cacc4684e67eabb409
md5deep	458b5ebc8c1bf7cacc4684e67eabb409
sha1sum	6beb9d5bfe512fdf34aa33f0c258a72caeb64995
sha1deep	6beb9d5bfe512fdf34aa33f0c258a72caeb64995
tigerdeep	21aca239cefd99d2f441eb3dd4576898961758292515897l
sha256deep	7bce318f4ce2833a02decbcde475c0b1164d1557567e55cc004f883276811d21
whirlpooldeep	d9ae3e0342558c1f1fc0eb5d8fcfcd97a4c932ddd869cba141d77367594660fb fefb292a895d97174757bbd85a1befe5d1d8e018b0f5d3bb0ceae05ded02f2f9

Summary

- Hash tools can give **identical hashes** to files with **different physical representations**
  - Hash tools at file system level ignore "sparseness"
- `du` detects file systems which support sparse files
  - FAT and Minix do not support sparse files
- When invoked at file system level, tools should warn the user about these possible inconsistencies

- Apart from challenging the integrity of the witness, is this a problem ... ?

Attack Scenarios

- Scenario 1: Insider wants to steal information
  - Insider prepares a large sparse file with the information
  - Insider copies the file to a small removable device
  - Insider replaces file with non-sparse file
  - Insider claims: "I didn't steal the file - it doesn't fit on my USB stick"
- Scenario 2: Insider wants to perform denial-of-service
  - Insider prepares a huge non-sparse file which consumes a lot of disk space
  - Disk space is exhausted, causing service disruption
  - Later insider replaces file with sparse version
  - Insider claims: "It's not my fault, I created a sparse file"
- In both cases, if only hashes at file system level are stored, attacker can tamper with evidence

References

- Brian Carrier: File System Forensic Analysis. Addison-Wesley, 2005.
- Eoghan Casey: Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet. Academic Press, 2. Ed., 2004.

- Harish Daiya: Sparse file testing images. Available online at `http://pi1.informatik.uni-mannheim.de/filepool/projects/hash-tool-testing/images.zip`