Comprehensive Black-box Methodology for Testing the Forensic Characteristics of Solid-state Drives

ACSAC
December 12th, 2013

Gabriele Bonetti, Marco Viglione, Alessandro Frossi, Federico Maggi and Stefano Zanero

Politecnico di Milano, Italy
Table of contents

1. Introduction
   Challenges in black box analysis and goals

2. Our testing methodology
   Overview
   Trimming
   Garbage collection
   Erasing patterns
   Compression
   Wear Leveling
   Files Recoverability

3. Putting it together: forensic friendliness

4. Discussion and conclusions
Introduction

Comprehensive Black-box Methodology for Testing the Forensic Characteristics of SSDs
SSD Technology

- NAND-based flash memory chips used as mass storage
- Increasingly popular as prices drop
- Widespread use in mobile devices
- On the surface, a snap-in replacement for rotational drives (HDD)
SSDs have a shorter lifespan as cells have a physical limit of approx. 10,000 program-erase cycles.

- **Rewrite** = blanking of a complete block (16 to 512kB)
- Led to development of flash translation layer (FTL) [5, 7], hw/sw combination that sits between ATA channel and memory chips.
FTL magic

- Write caching
- Trimming
- Garbage collection
- Data compression
- Data encryption/obfuscation
- Bad block handling
- Wear leveling
In other words...

- In HDDs we can reliably physically address a sector from the OS and read it on the drive.
- In SDDs FTL translates logical block addresses (LBA) as requested by the OS into the respective physical block addresses (PBA) on memory chips. The underlying mapping is completely transparent and can be modified by the FTL at any time for any reason. The FTL may move data around or blank data even if the OS is not running.
- Yay. Most forensic approaches and tools rely on the ability of the OS to access the raw data on the disk.
Can we bypass the FTL?

- Not via software.
- In theory, can be bypassed by reading directly the memory chips [3] (via flashing tools, JTAG port, or physical extraction of chips)
- [4, 8] built a complete custom setup to interact with flash memory chips using an FPGA and custom wing boards. Their goal is prototyping but similar setup could be used to reimplement FTL logic and read memory chips
- In any case:
  - extremely time and money consuming process (needs custom hardware, reverse engineer FTL implementation...)
  - non-repeatable, leads to alteration or destruction of the evidence
  - very brittle process depending on firmware, hardware...
  - information not public and actually heavily protected IP of vendors
Challenges in black box analysis and goals
An unclear picture

- Previous work suggests impact on black box forensic analysis.
- [2] analyzes file recovery rate (SSDs vs HDDs)
  - Observes that even a write blocker does not prevent the FTL from modifying and in some cases blanking the evidence
  - Suggests a filesystem aware garbage collection feature in FTLs
- [6] tested 16 SSDs with usage scenarios and concluded that different combinations of usage, OS and file size influence recoverability (most sensible and extensive paper to date)
- [1] found that carving didn't work at all on SSDs
- Other (non-scientific) reports suggested that data duplication due to wear leveling would increase recoverability
The need for a triage methodology

- We developed a simple and affordable black-box triage procedure to:
  1. assess impacts of FTL on the use of black-box tools
  2. assess likelihood of success of a white-box attempt
- Test driven workflow of experiments to assess the behavior of the FTL under different conditions
- We can determine whether a SSD implements trimming, garbage collection, compression and/or wear leveling
Our testing methodology
Overview
What we test for (1)

**TRIM:** preemptive blanking of erased blocks marked for trimming by the OS. Negative impact on forensics as data persistence is reduced. [2] notes that this can occur even with a write blocker, impacting acquisition. Our methodology can determine the percentage of blocks that get erased and how fast.

**Garbage collection:** hypothesized by [2] to work with a filesystem-aware controller, that TRIMs block without OS support. Forensic impact obvious. We can determine whether it is employed by the SSD under examination.

**Erasing patterns:** peculiar behaviors shown by some SSDs when using TRIM.
What we test for (2)

**Compression**: transparently employed by some drives to use less physical blocks and reduce wear. No challenges in black box analysis, but definite challenge in white box. We can verify whether compression is active.

**Wear leveling**: spreads consumption of cells as evenly as possible across the drive. We test for the so-called “write amplification” effect, which is a direct consequence of the wear leveling.

**Files recoverability**: a test on the efficacy of black box file recovery techniques.
Our test drives

<table>
<thead>
<tr>
<th>SSD</th>
<th>WL</th>
<th>TRIM</th>
<th>GC</th>
<th>Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corsair F60</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Samsung S470</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Crucial M4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

**Table**: Test drives, and their features as reported by vendors.
A small but important caveat

- SSDs are equipped with a small amount of DRAM-based cache memory to reduce physical writes.
- This can bias any test using small files (i.e. smaller than 512MB-1GB, typical cache size).
- Experiments in [1] were probably biased by this, and files were never written to disk, explaining zero carving results. Ditto for [6].
- Solution is simply to disable cache (e.g. on Linux via `hdparm -W 0`) or to use large files.
Trimming
Methodology

- Disk formatting
  - NTFS or ext4
- Disk filling at different percentages
  - 25%, 50%, 75%, 100%
- Start real-time analysis of disk zeroed space
- Quick format in OS with TRIM support
  - Win7 for NTFS
  - Ubuntu for ext4
- Single files deletion
- Check state of deleted file's sectors
  - `hdparm --fibmap <filename>`
  - `hdparm --read-sector <address> /dev/sdx`

Figure: TRIM test flow
Results

• If TRIM present and active, it activates in 1-10 seconds
• On NTFS, Samsung S470 and Crucial M4 trimmed aggressively, wiping the disk/file in under 10 seconds
• Weird behavior of Corsair F60 as in figure: erased blocks someway proportional to used space. Some files wiped in at most 3 seconds after deletion, others untouched

• ext4 all disks erased in about 15 seconds with format. Samsung S470 did not erase on file delete. Crucial M4 was notified of TRIM only on unmounting. Corsair F60 erased all files “correctly”
Garbage collection
Methodology

Disk filling at different percentages 25%, 50%, 75%, 100%

if not already in use, switch to a forensic OS

Start real-time analysis of disk free space

Quick format

Keep disk active for a long time, overwriting files

Keep disk idle for a long time

16 h

Periodic verification of free space percentage and zeroing time

Figure: Garbage collection test flow.
Results

- [2] found out that GC triggers in almost 3 minutes.
- non-authoritative sources state 3 to 12 hours
- In our test, none of the SSDs performed garbage collection. We even tried to replicate the exact test of [2], with identical hardware, software and firmware version, but to no avail, even with the assistance of the author.
Erasing patterns
Methodology

Figure: Erasing patterns test flow.

Stefano Zanero

Comprehensive Black-box Methodology for Testing the Forensic Characteristics of SSDs
Results

- Certain SSD controllers may exhibit unexpected trimming patterns
- In our case, target of interest was the Corsair F60 SSD. See maps below:

(a) 10%
(b) 50%
(c) 75%
(d) 100%

- Validated on file recovery. Files in green stripes are recoverable only 0.34% of the times, outside 99%
Compression
Methodology

OS write cache disabling

Creation of data blocks of equal size 10 GB

/dev/zero

Creation of low entropy data

Copy data in main memory

Write time calculation

/writestat

Copy data in main memory

Write time calculation

/dev/urandom

Creation of high entropy data

Copy data in main memory

Write time calculation

Write time comparison

Figure: Compression test flow.
Figure: Mean and variance of the sampled throughput among 15 repeated transfers of 10GB low (top) and high (bottom) entropy files. Intuition is that overhead for hw compression negligible, thus takes less to write files that can be compressed. Samsung and Crucial drives show no difference: no compression; Corsair performs hw compression instead.
Wear Leveling
Methodology

- Partial disk filling at different percentages (25%, 50%, 75%)
- Start real-time analysis of disk free space
- Creation of a file with known pattern
- Multiple overwrites of file with known pattern
- OS write cache disabling
- Periodic verification of zeroed space
- Disk image acquisition
- Carving to find multiple copies of known file

Figure: Wear leveling test flow
Results

- Not test for presence (almost default) but for usefulness for forensic analysis
- From black box PoV, if write amplification does not happen, or is completely masked, there is no difference
- No drives showed write amplification from an external PoV
Files Recoverability
Methodology

Choice of a file easy to recover (to eliminate weight of carver errors)

Disk filling

Quick format

Disk image acquisition

Carving

Recovered files hash calculation

File hash calculation

Integrity check on recovered files

md5sum <file>

sha1sum <file>

Figure : Files recoverability test flow.
### Results

<table>
<thead>
<tr>
<th>SSD</th>
<th>FS</th>
<th>Written</th>
<th>Recovered</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung</td>
<td>NTFS</td>
<td>112,790</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>ext4</td>
<td>110,322</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Corsair</td>
<td>NTFS</td>
<td>101,155</td>
<td>71,607</td>
<td>70.79%</td>
</tr>
<tr>
<td></td>
<td>ext4</td>
<td>99,475</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Crucial</td>
<td>NTFS</td>
<td>112,192</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>ext4</td>
<td>110,124</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>

**Table**: Files recoverability test results: the drives implementing an aggressive version of TRIM (Samsung S470 on NTFS and Crucial M4), did not allow the recovery of any file after a format procedure. The Corsair F60 on NTFS, as expected, has a non-null recovery rate due to the erasing pattern its TRIM implementation exposes. On ext4, however, this same disk allowed the recovery of 0 out of 99,475 files.
Putting it together: forensic friendliness
Methodology

4.1 TRIM

4.2 Garbage Collection

Zeroing detected?

Yes

Yes

Yes

4.6. Files Recoverability (single file removal)

4.6. Files Recoverability (quick format)

4.3 Erasing Patterns

Intact files recovered?

Intact files recovered?

Intact files recovered?

None

Some files

Antiforensics feasible

Class B

Class C

Class D

Figure: Use case workflow for assessing the forensic friendliness of a SSD.
Discussion and conclusions
Limitations

- We did not test multiple firmware versions, as firmware upgrades are one-way in most cases and this would make the experiments difficult to repeat.
- We did not test on difference of device driver and AHCI commands, for simplicity.
- If investigator does not know OS version in use, our methodology may not give usable insights.
• SSDs implement techniques that are potentially disruptive to black box forensics
• We created a triage workflow to understand impact and potential gain of white-box approach
• We showed that the combination of controller, OS, filesystem and even disk usage can deeply influence forensic procedures
• We showed that garbage collection is not currently offered by leading drives on the market
Thank you for your attention. **Questions?**

Let’s keep talking on Twitter (@raistolo) or on email (stefano.zanero@polimi.it)
The work has been partially funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 257007 “SysSec”
References I


References II


Michael Wei, Laura M. Grupp, Frederick E. Spada, and Steven Swanson.

Reliably erasing data from flash-based solid state drives.