Comments on “Digital Investigation Techniques: A NIST Scientific Foundation Review” (NISTIR 8354-DRAFT)

Glossary and Acronyms

Page	Text	Comments
iv	**Advanced Format** Created to address technical issues with the 512-byte storage device sector size by changing storage device sector size from 512-bytes to a multiple of 512-bytes such as 4096-bytes, i.e., storage devices with a sector size larger than 512-bytes.	Not all storage devices with a sector size larger than 512 bytes are *Advanced Format* (e.g., DVDs are not *Advanced Format* despite their 2048-byte sector size).
v	**Disk Imaging** The process of acquiring the digital contents of a storage device (fixed disk, removable disk, flash drive, etc.). This acquires all the data on a device including files, metadata, and contents of unallocated areas of the device.	Usually, disk imaging involves copying data which is exposed to a host (but not all exposed data is often copied – e.g., S.M.A.R.T. values can be skipped). Some data stored on a device is not exposed to a host (this includes raw data from service areas and blocks used for overprovisioning), thus not copied by most disk imaging tools (it is impossible to read such data using standard ways of communicating to a storage device). The definition can be changed to reflect this.
vi	**File System** A method for organizing files on a storage device. Common file systems on Windows systems are NTFS, ExFAT and FAT. LINUX systems use ext4 and FAT. Apple Macs use HFS+, APFS, FAT and ExFAT.	Ext2 is still popular in the Linux world (it is often used for volumes that do not require journaling, like “*/boot*”).

vi	**Fixed media** A storage device that is physically installed in a computer.	1. Two definitions (of “Fixed media” and “Removable media” respectively) are not clear enough. According to them, it is possible for removable media to become fixed once it is attached to a computer port (it becomes *“physically installed in a computer”*). 2. Are all hot-pluggable drives removable? Are USB flash drives removable? Are USB HDD/SSD enclosures removable? Although many types of storage devices can be hot-plugged, not all of them are considered as removable (e.g., SATA HDDs). The current definitions (of “Fixed media”, “Removable media”, and “Storage Device”) do not cover this case. 3. Also, there is an issue with USB devices. USB flash drives are removable, because they can be attached and removed while the computer is running. But USB flash drives have non-removable media, because they do not contain swappable memory chips. Even most USB HDD/SSD enclosures have non-removable media, because their internal drives cannot be swapped on the fly (without resetting the enclosure, so they require an initiator-target nexus loss event). As a result, USB flash drives must report their removable medium bit as 0 (but most of them do not). But if this bit is correctly set to 0, some popular operating systems treat such a storage device as non-removable (see SanDisk Answer ID 12830: https://kb.sandisk.com/app/answers/detail/a_id/12830), because they see no difference between removable devices with non-removable media and devices with removable media. For further discussion, see “SPC-6: Removable Medium Bit Expectations” (https://www.t10.org/cgi-bin/ac.pl?t=d&f=20-082r1.pdf). So, the current definitions can be adjusted to define removable (and non-removable) storage devices, as well as removable (and non-removable) media.
ix	**Removable media** A storage device that is either (1) a data container that is inserted and removed from a data reader or (2) a storage device that can be connected or removed from a computer while the computer is running.	
ix	**Storage Device** *[…]* * Fixed media physically installed in a computer. The computer must be powered off to install or remove the storage device. * Removable media. Can be installed or removed while the computer is running. Small storage devices are called flash drives or thumb drives (they are about the size of a human thumb). These devices are usually connected via a USB interface. *[…]*	
viii	**Metadata** Metadata is a description of stored data. Categories of metadata	There are more metadata types than listed in the definition (e.g., RAID metadata – this type of metadata describes how to assemble the array of physical drives to get one or more virtual drives).

	include: (1) application metadata (in a document this could be author, organization, etc., in a database such as SQLite there is metadata to describe the layout of the stored data within the database), (2) file system metadata (placement of the file within the file system, owner, permissions, MAC times, etc.), (3) partition metadata that identifies the type of file system the partition contains and global file system parameters, and (4) device metadata describes the layout of partitions on a device.	So, the phrase *“Categories of metadata include”* can be changed to *“Categories of metadata include, for example”* (or to something similar).
viii	**NTFS** New Technology File System. Microsoft Windows file system introduced in 1993, revised several times over the years.	The NTFS file system has been revised many times, so the word “several” is misleading (“multiple” is more suitable). Modern revisions do not increment the format version number, though. For example, the following features have been implemented in the past years (all of them affect the on-disk format, but do not change the version number of the whole file system): - energy-efficient metadata logging (requiring less writes per logged operation); - per-directory case sensitivity; - reparse point security (trust levels); - storage reserve (reserved file system areas and file tags); - very large clusters (up to 2M bytes).
viii	**Operating System** The software that creates the digital environment for running software on a computer or other digital device. Most operating systems are variants of either MS Windows (95, 98, 2000, Vista, XP,	Although many operating systems have gone away, the phrase “[m]ost operating systems are variants of *[two operating system families]*” is still not suitable here. Embedded systems, feature phones, various hardware devices often use other operating systems (e.g., Azure RTOS ThreadX and MOCOR).

	10, etc.) or UNIX (BSD, Linux, Mac OS, iOS, etc.).	
viii	**Partition** A contiguous area of a storage device used to contain a formatted file system.	Partitions do not necessarily contain a formatted file system. Other types of data that can be stored in a partition: - a swap space (occupying a whole partition); - a software RAID volume (which provides a layer between a partition on a physical drive and a file system) or a volume created using similar technology (e.g., Linux LVM). Operating systems can reserve some disk space for their internal use by creating an unformatted partition. Also, operating systems can merge two or more partitions into one virtual partition (by providing a way to split data between multiple existing partitions), this is another layer between a partition on a physical drive and a file system.
viii	**Partition Table** A table describing the layout of a physical storage device that has been divided into partitions, each partition contains a separate file system.	
ix	**Storage Device** An electronic or optical device that can store data for later retrieval. *[…]*	This definition does not include magnetic tape cartridges (they are still used for backups).
ix, x	**UICC card** A Universal Integrated Circuit Card (also called a SIM card) contains phone number and account information for mobile devices. An integrated circuit card that securely stores the international mobile subscriber identity (IMSI) and the related cryptographic key used to identify and authenticate subscribers on mobile devices.	Not all SIM cards store a phone number. Also, “UUIC card” stands for “universal integrated circuit *card card*”.
x	**Write Blocking** Techniques designed to prevent any modification to digital	The write blocking techniques cannot prevent modifications performed by a storage device itself. So, write blocking does not prevent “any modification”.

	media during acquisition or browsing.	For example, write blocking does not stop file-system-aware SSDs from reclaiming unallocated space of supported file system types. Also, write blocking does not prevent modifications not exposed to a host (like wear leveling, which changes the layout of data on the physical layer, while data returned to a host remains the same).

Executive Summary

Page	Text	Comments
4	11. **KEY TAKEAWAY #4.6**: It is not feasible to test all combinations of tools and digital evidence sources.	It is practically impossible to test all combinations of tools and source data (different file system types and their states on storage devices, as well as different types of underlying media). But different types of storage devices can produce unexpected inconsistencies during acquisitions. So, various types of storage devices have to be carefully selected for a specific test (e.g., both HDDs and SSDs should be tested against an acquisition tool). *More information about this takeaway is provided below (see page 12 of this document).*
4	12. **KEY TAKEAWAY #4.7:** Extensive tool testing of over 250 widely used digital forensic tools showed that most tools can perform their intended functions with only minor anomalies.	It is unclear what constitutes a minor anomaly. Some clarification is needed in the executive summary. *More information about this takeaway is provided below (see page 19 of this document).*

Chapter 2

Page	Text	Comments
11	*[…]* and each partition is formatted with a selected file system including layout of file placement on the device	A structure describing which data blocks are used by a given file belongs to file system (volume) metadata; this is not partition metadata. Partitions should not be confused with file systems (volumes), a partition can contain a file system, but it is not equal to a file system (also, it is possible to create a file

	in partition metadata, and any stored data.	system on a storage device without using a partition table, such a file system starts at LBA of 0).

14, 15	**2.3 Time**
	
	Times and dates can often exhibit subtle nuances that are prone to misunderstanding.
	[…]
		There are several remarkable issues worth mentioning.
		
		1. Local timestamps can be ambiguous, because two different UTC timestamps can point to the same local date and time values due to time folding (when clocks go back).
		
		Here is an example:
		
		`$ TZ='Europe/Stockholm' date -d @1667092000 '+%F %T'`
		`2022-10-30 02:06:40`
		`$ TZ='Europe/Stockholm' date -d @1667088400 '+%F %T'`
		`2022-10-30 02:06:40`
		
		Printing the time zone explicitly can help mitigate the problem:
		
		`$ TZ='Europe/Stockholm' date -d @1667092000 '+%F %T %Z'`
		`2022-10-30 02:06:40 CET`
		`$ TZ='Europe/Stockholm' date -d @1667088400 '+%F %T %Z'`
		`2022-10-30 02:06:40 CEST`
		
		Still, there are cases when clocks go back without changing the time zone identifier.
		
		2. Time zones are affected by political issues. People living in a specific area can use a different time zone than specified officially. Or two time zones can coexist.
		
		Also, applying time zone offsets to timestamps in the future is dangerous (nobody knows which time zone offset will be in effect by the time).

16	For example, a partition table is a configuration file that describes the layout of a storage device.	A partition table is not a configuration file. Popular partition table formats (e.g., MBR and GPT) store their metadata in blocks (sectors) outside of any file systems (and, thus, outside of files).

18	LINUX systems use ext4 and FAT.	Ext2 is still popular in the Linux world.
20	*[…]* TRIM for SATA devices *[…]*	The Trim command is not specific to SATA drives, it could be sent to USB drives too (if the SCSI ATA PASS-THROUGH command is supported by a USB drive or an SSD enclosure).
23, 24	The primary sources of knowledge about digital forensic techniques are the following: * Vendor-Independent Forensic Technique training classes * Tool Vendor offered classes * Forensic Tool Vendor white papers and other support documents * Forensic Professional Organizations * Standards Organizations * Online training videos * Blog Posts * Academic Peer Reviewed papers in Conferences and Journals * Academic course work * Reference books * Operating system and computer hardware vendors support documents * Reverse engineering of software: Operating system, file system or application	1. There are more primary sources of knowledge: - source code of operating systems, drivers, applications (including leaked source code), debugging symbols (the latter partially overlaps with reverse engineering, though); - podcasts; - talks at conferences. 2. On page 23, “[o]nline training videos” are listed. On page 27, this item is called “[o]nline [v]ideos”. Based on the context, “online videos” is a better description. *Similarly, there are other items called differently in the list (on pages 23-24) and in the description (on pages 24-29), they will not be explicitly mentioned here.*
27	**2.8.6 Online Videos**	
29	**2.8.11 Software Developer Documentation** Documentation about operating system internal organization is often available and provides a rich source of information about trace artifacts	It should be noted that such documentation can be misleading, wrong, or out of date. Software developers can forget to update the documentation once the implementation has changed. Or software developers can document the design (or even the design expectations), but not the actual implementation.

	that may be of forensic value. Documentation of individual applications and sometimes the source code of the applications might be available.	

Chapter 4

Page	Text	Comments
34	The acquired data is placed into a container file that represents the acquired data.	It is possible to clone a storage device. In this case, data from a source storage device is copied to a destination storage device directly (no file containers are used). Data on the source storage device located at LBA of 0 will be copied to LBA of 0 on the destination storage device, data on the source storage device located at LBA of 1 will be copied to LBA of 1 on the destination storage device, etc.
35	Whenever possible, acquisition should be done in conjunction with either a hardware write blocking device or a software write blocking tool to avoid modification of the original data.	There is an approach called “quasi software write blocking” – when an operating system is built not to write to attached storage devices. Examples of such operating system builds include Windows FE and Helix3 Pro. In this case, there is no component that inspects I/O requests to block unwanted writes (like a write blocking driver), but the operating system is designed to work in the “read-only” mode (most Linux distributions fail here, but the concept still exists). (More information can be found here: https://github.com/msuhanov/Linux-write-blocker/tree/master/research.)
35	**4.2.2 Mobile Device Acquisition** For mobile device forensics, there are many considerations and options for acquiring and analyzing data from a mobile device (SWGDE 2016a, 2016b, 2019b):	1. File system acquisitions can be full (all allocated files and directories are copied when a mobile device is rooted or jailbroken, either temporarily or permanently) or partial (many files and directories are not copied in this case because of security permissions). 2. Many mobile devices support memory cards. Memory cards can be acquired too (typically, they are

	* Logical acquisition: *[…]* * Selective acquisition: *[…]* * File system acquisition: *[…]* * Physical acquisition: *[…]* * Universal Integrated Circuit Card (UICC), also called a Subscriber Identity Module (SIM Card) acquisition: *[…]*	removed from a mobile device and imaged separately). This option can be added to the list.
36	After digital data has been acquired to an image file it needs to be verified that the acquired data has not been changed. Cryptographic hashing is used to detect inadvertent or deliberate changes.	Cryptographic hashing alone does not prevent deliberate changes that involve updating the hash value to reflect the changed data (i.e., it is possible to change the acquired image file, calculate the hash over altered data, and then specify it as the original hash value recorded during the acquisition, or it is possible to alter data on a source storage device and then acquire it again). This is why cryptographic hash values should be recorded in a way that protects them from alteration (e.g., by maintaining a chain of custody or by using digital signatures). Simply storing cryptographic hash values along with acquired data is not enough (this is why evidence containers like E01 are not “self-authenticating”, they only protect against inadvertent changed and data transfer errors).
37	If the deleted data has been overwritten or allocated to a new object, the deleted data cannot be recovered.	This is true, but such a deleted object can be found in another source (e.g., in a volume shadow copy).
43	The general validation and verification for a given version of a tool can be done once. It does not need to be performed by every lab.	1. Software tools depend on the environment (the operating system, shared libraries, and their configuration), so a tool running in more than one environment can produce different results. Some examples are: - the same UTC timestamp can be converted to the same local time zone (and vice versa) differently depending on the time zone library and its version; - the same Unicode string (like a file name) can be displayed (reported) differently depending on the Unicode support implemented in the operating system (a list of supported Unicode characters can vary);

		- a tool (and, in general, a technique) relying on file system and volume management drivers can produce different results depending on the versions of these drivers and the version of the operating system; - similarly, a tool using shared libraries to parse data can produce different results depending on the version of various third-party libraries and their data (especially, when on-disk data format updates are not backward-compatible). And more specific examples are: - Tools that process data from volume shadow copies by mounting the disk image in the Windows operating system and then using the devices exposed by the volume snapshot driver can see no volume shadow copies when running under Windows 11 (the volume snapshot driver does not expose volume shadow copies if the underlying storage device is read-only, this problem does not exist previous versions of Windows); - Tools that parse Windows event logs by using API functions provided by the Windows operating system can display the same event (in the human-readable form) differently depending on the operating system version used (because string templates used in different versions of the operating system can vary). This also affects event logs created by third-party applications. This why validation for a given version of a tool should be done more than once in order to cover more environments in which this tool is used. This requirement can be relaxed (within reasonable limits) for tools that do not parse data, for tools that are included into “stable” environments (e.g., live distributions and kiosks), and for specific functions of tools that are known to be “stable” across different environments (e.g., using a bundled third-party library). 2. Some software can be updated without changing its version number. In 2016, PassMark released an updated version of the OSFClone tool (this is a live distribution used for acquisitions), which included a configuration change to disable the swap space activation on attached storage devices. The version number of the tool did not change, though. Another example is software packaged by many Linux distributions. This software is often patched to

introduce distribution-specific changes (fixing compiler warning, typos, vulnerabilities, and even some other bugs), while the version number remains the same (only the revision number is changed, which is a distribution-specific field, often not reported by a tool when displaying its version).

A more specific example is the dcfldd tool. In one version of the Debian operating system, the following version of the tool is shipped: 1.3.4.1-10, but the tool reports its version as 1.3.4-1 (the revision number, 10, is omitted). This revision includes multiple patches, see the Debian changelog: https://metadata.ftp-master.debian.org/changelogs//main/d/dcfldd/dcfldd_1.3.4.1-10_changelog. Along those patches, there is one moving the storage device size probing function to a newer, 64-bit, call (“10_fix-probing-of-large-block-devices.patch”), thus *introducing* one bug that can be encountered when testing the patched revision of the tool (but both revisions, patched and unpatched, report the same version number).

With the patch applied, the tool gives the following output:

```
# dcfldd if=/dev/sda of=/dev/null sizeprobe=if
[0% of 249511424Mb] 92160 blocks (2880Mb) written. 24:03:54 remaining.^C
92204+0 records in
92204+0 records out
```

Without that patch, the output is:

```
# dcfldd if=/dev/sda of=/dev/null sizeprobe=if
[0% of 487327Mb] 96000 blocks (3000Mb) written. 00:00:00 remaining.^C
96082+0 records in
96082+0 records out
```

The real size of the block device (“/dev/sda”) is, in bytes:

```
# blockdev --getsize64 /dev/sda
510999396352
```

Or 487327 MiB. The patched revision of the tool report an invalid size. The reason is that the patched revision takes the device size, in bytes, and multiplies it by the sector size, in bytes

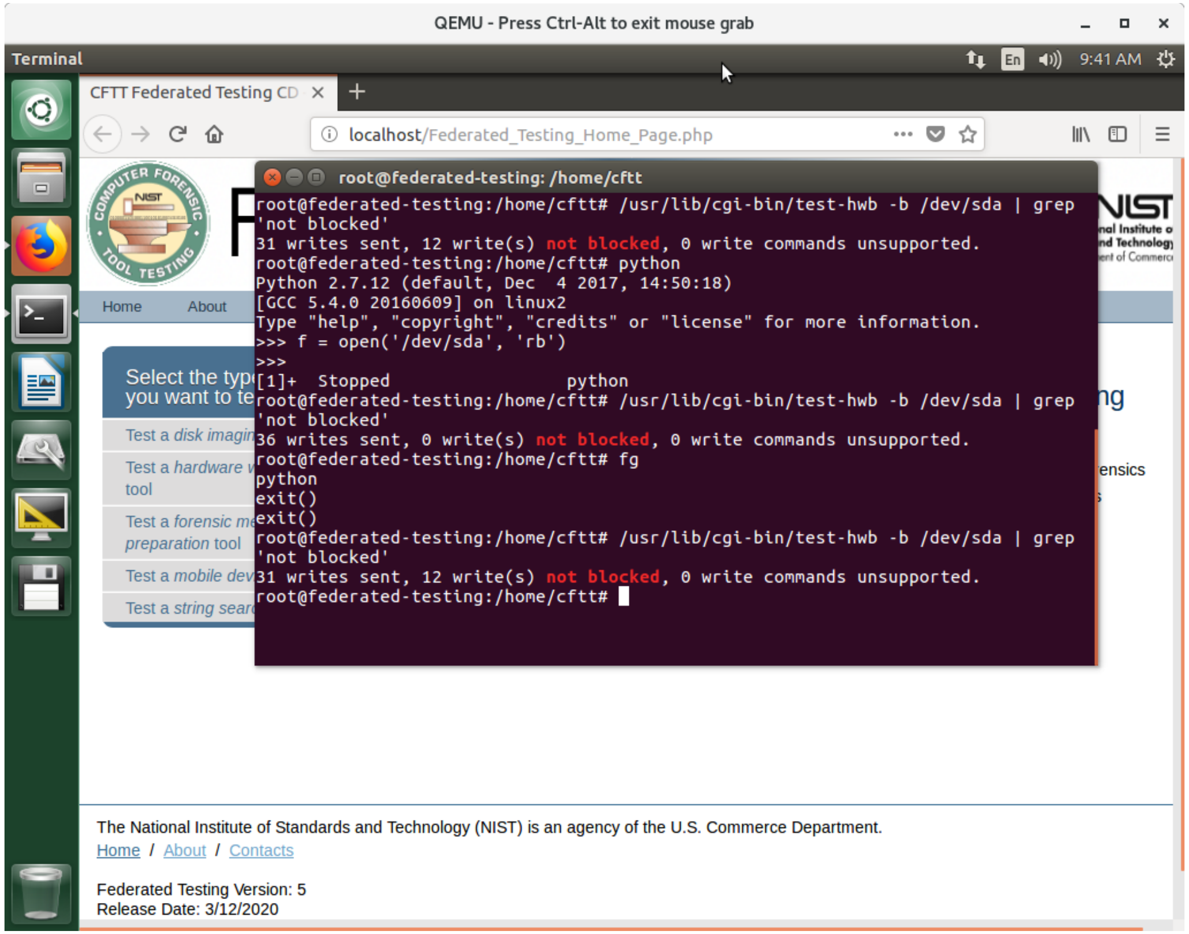
		(249511424÷487327=512). This is why validation must account possible code and configuration changes not affecting the version number.
47	**KEY TAKEAWAY #4.6:** It is not feasible to test all combinations of tools and digital evidence sources.	It is practically impossible to test all combinations of tools and source data. A single file system type has many thousands of possible states (this includes all flags stored in the file system header and in core metadata locations, all possible states of low-level structures, and all types of underlying media), there is no way to test every state against a selected tool, while testing only those file system states that are considered “common” or “real-world” can result in important issues being undetected. A volume formatted using the NTFS file system can be in the following states: a.1. unmounted properly (clean); a.2. not unmounted properly, without an ongoing I/O operation interrupted (when disconnecting the volume); a.3. not unmounted properly, with an ongoing I/O operation interrupted; b.1. not marked as corrupt; b.2. marked as corrupt; c.1. starting at the first block (sector) of a partition; c.2. starting at LBA of 0 (no partition table is used on a storage device, this is often found on USB flash drives); d.1. located on the SSD; d.2. located on the HDD; d.3. located on the USB flash drive; e.1. using the $LogFile version 1.1; e.2. using the $LogFile version 2.0; f.1. using 1024-byte file record segments; f.2. using 4096-byte file record segments. Although the list above is incomplete, it gives dozens of possible combinations to test. It is possible to

		test each state in the corresponding group separately, but this will never produce complete results, because some issues can be uncovered only when a specific combination of states is encountered. Here are two examples: - It is important to run tests using, at least, HDDs and SSDs. BitCurator is a distribution used for archival and digital forensics acquisitions, it can be installed on a physical machine. If an SSD is attached as a secondary drive and its file system is mounted in the read-only mode for data preview, the unallocated space of this file system can be trimmed on Sundays or Mondays. See the corresponding issue: https://github.com/BitCurator/bitcurator-distro-main/issues/102. - It is important to account complex file system states when testing data acquisition tools and hardware write blockers. Tableau TD3 is a forensic imager and a network-based write blocker, it has several ports marked as write-blocked (including SATA and USB). When a storage device is attached to a “write-blocked” port, a write command can be sent to it. All of the following conditions must be met to trigger the issue: a. a source drive has at least one Ext4 volume supported for data preview (e.g., this is an HDD with a partition table and its only partition is formatted using the Ext4 file system); b. an error (e.g., an I/O error) is recorded in the file system journal; c. no error is recorded in the file system superblock. (These conditions can be encountered in the real world, but they are unlikely to be selected during a usual validation test.) Under these conditions, the Ext4 driver of the Tableau TD3 device (its firmware is Linux-based) will transfer the error code from the journal to the superblock, thus issuing a write command through a “write-blocked” port. The problem arises from the fact that the device implements no write blocking and it

		automatically mounts a file system for data preview (later, a user can browse this file system using a web browser; the file system is mounted automatically, even when no network-based functions are requested by a user), so the read-only mode is not enforced. See the corresponding paper: https://github.com/msuhanov/Linux-write-blocker/tree/master/research. And a test image (to be written to a partition of an HDD to be tested): https://github.com/msuhanov/articles/blob/master/misc/ext4.raw. This problem was discovered in 2016, but the most recent firmware (version 2.1.1) is still affected. The problem does not affect Tableau TX1 (a more recent product). This is why validation requires prior reverse engineering to locate weak spots (instead of blindly testing thousands of file system states, which is practically impossible).
50	Tools often omit readable sectors surrounding a bad sector, usually related to how the file system blocks disk sectors for the interface (USB, SATA, Firewire, etc.) used to access the hard drive.	1. Disk imaging tools skip readable sectors surrounding an unreadable one because of the error granularity. This has nothing to do with file systems (although file systems can track unreadable sectors using larger blocks like clusters, this has no direct relation to disk imaging). For example, if a disk imaging tool reads 8 sectors at a time and one of them is unreadable, the whole read request fails (as a result, all of 8 sectors are reported as unreadable). In this situation, a disk imaging tool can read the failed range again, requesting 1 sector at a time. If the tool does not do that, an anomaly is observed (not all readable data is acquired). 2. Hardware write blockers can affect the error granularity. For example, Tableau T356789iu is a forensic bridge that can report 128 sectors as unreadable when only one unreadable sector is encountered (and there is no way to read those 127 sectors from the host side). This problem does not affect the latest firmware (the only firmware version affected is 1.3.0). The problem arises from the fact that the bridge provides a read-ahead cache, which is kept in its random-access memory. All read requests coming from the host are served through this cache. And the cache is populated with a poor error granularity.

50	The most serious failure ever observed was reported in 2003 for SafeBack Version 2.0 (National Institute of Justice and National Institute of Standards and Technology 2003). *[…]*	There was another failure of equal value observed. In the test results for the dcfldd tool (version 1.3.4-1), there is a case where data on a destination storage device becomes misaligned after an unreadable sector is encountered on a source storage device. According to the report (https://www.dhs.gov/sites/default/files/publications/DCFLDD%201%203%204-1%20Test%20Report_updated.pdf): When a drive with faulty sectors was imaged (test case DA-09) the data cloned to the target drive became misaligned after faulty sectors were encountered on the source drive. For example, sector 6,160,448 on the target drive contained the contents of sector 6,160,392 from the source, sector 6,160,449 on the target contained the contents of source sector 6,160,393, and so on. The size of the offset or misalignment between the data on the source and target drives grew as more faulty sectors were encountered on the source. This means that the dcfldd tool produces inaccurate results for drives containing unreadable sectors. Since unreadable sectors can occasionally be found on HDDs and SSDs, as well as on USB flash drives and memory cards, the tool is unreliable and should not be used for acquisitions. Otherwise, misaligned data would be extremely hard to account (e.g., a tool parsing a file system does not expect a data block to be stored at a different sector number). More information about the issue can be found here: https://github.com/adulau/dcfldd/issues/1.
51	Except for one model device, hardware write-block devices always blocked write commands. The firmware for the one blocker that allowed write commands was quickly fixed by the vendor.	1. There was another failure observed. Coolgear SS-127ASD is a SATA/PATA-to-USB adapter with the write blocking switch. A similar piece of hardware is also sold under other brands (e.g., AgeStar). According to three reports, this device does not block some write commands:

50	Most write blockers on the market were able to block commands that would have changed a drive. The few exceptions were for uncommon commands or, in one case, where a vendor was unaware of a change to a chipset (that was quickly fixed).	https://www.dhs.gov/sites/default/files/publications/testresults_for_cool_gear_ss-127asd_sata-ide_adapter.pdf, https://www.dhs.gov/sites/default/files/publications/testresults_for_cool_gear_sata-ide_adapter_windows.pdf, and https://www.dhs.gov/sites/default/files/publications/testresults_for_cool_gear_sata-ide_adapter_linux.pdf. 1.1. On page 50, one more issue with hardware write blockers is mentioned. 2. It should be noted that such tests utilize the “read-write-read-compare” approach (a sector is read to get its original data, then an attempt is made to write to that sector, and the sector is read again to see if anything has changed in its data), or a similar approach. In theory, a hardware write blocker can fail to block a write command, but an attempt to read a successfully overwritten sector would produce old (intact) data for that sector (while data on the storage device has changed), because all read requests are served through a read-ahead cache, which is not invalidated on that write operation. To account such cases, a hardware write blocker should be rebooted before checking the sectors for possible data modifications (to reset its random-access memory and, therefore, the cache). A similar issue affecting the “test-hwb” tool was reported to NIST in 2018 (this issue has a similar nature, but it occurs on the host side). The tool fails to detect modifications on a storage device if a corresponding block device is opened in another program (like a partition table viewer). The issue is still present in the Federated Testing suite, version 5:


QEMU - Press Ctrl-Alt to exit mouse grab
Terminal
En
9:41 AM
CFTT Federated Testing CD
localhost/Federated_Testing_Home_Page.php
root@federated-testing: /home/cftt
Home
About
Select the type you want to te
Test a disk imagin
Test a hardware tool
Test a forensic me preparation tool
Test a mobile dev
Test a string sear
ng
ensics

```
root@federated-testing:/home/cftt# /usr/lib/cgi-bin/test-hwb -b /dev/sda | grep
'not blocked'
31 writes sent, 12 write(s) not blocked, 0 write commands unsupported.
root@federated-testing:/home/cftt# python
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> f = open('/dev/sda', 'rb')
>>>
[1]+  Stopped                 python
root@federated-testing:/home/cftt# /usr/lib/cgi-bin/test-hwb -b /dev/sda | grep
'not blocked'
36 writes sent, 0 write(s) not blocked, 0 write commands unsupported.
root@federated-testing:/home/cftt# fg
python
exit()
exit()
root@federated-testing:/home/cftt# /usr/lib/cgi-bin/test-hwb -b /dev/sda | grep
'not blocked'
31 writes sent, 12 write(s) not blocked, 0 write commands unsupported.
root@federated-testing:/home/cftt#
```

The National Institute of Standards and Technology (NIST) is an agency of the U.S. Commerce Department.
Home / About / Contacts

Federated Testing Version: 5
Release Date: 3/12/2020

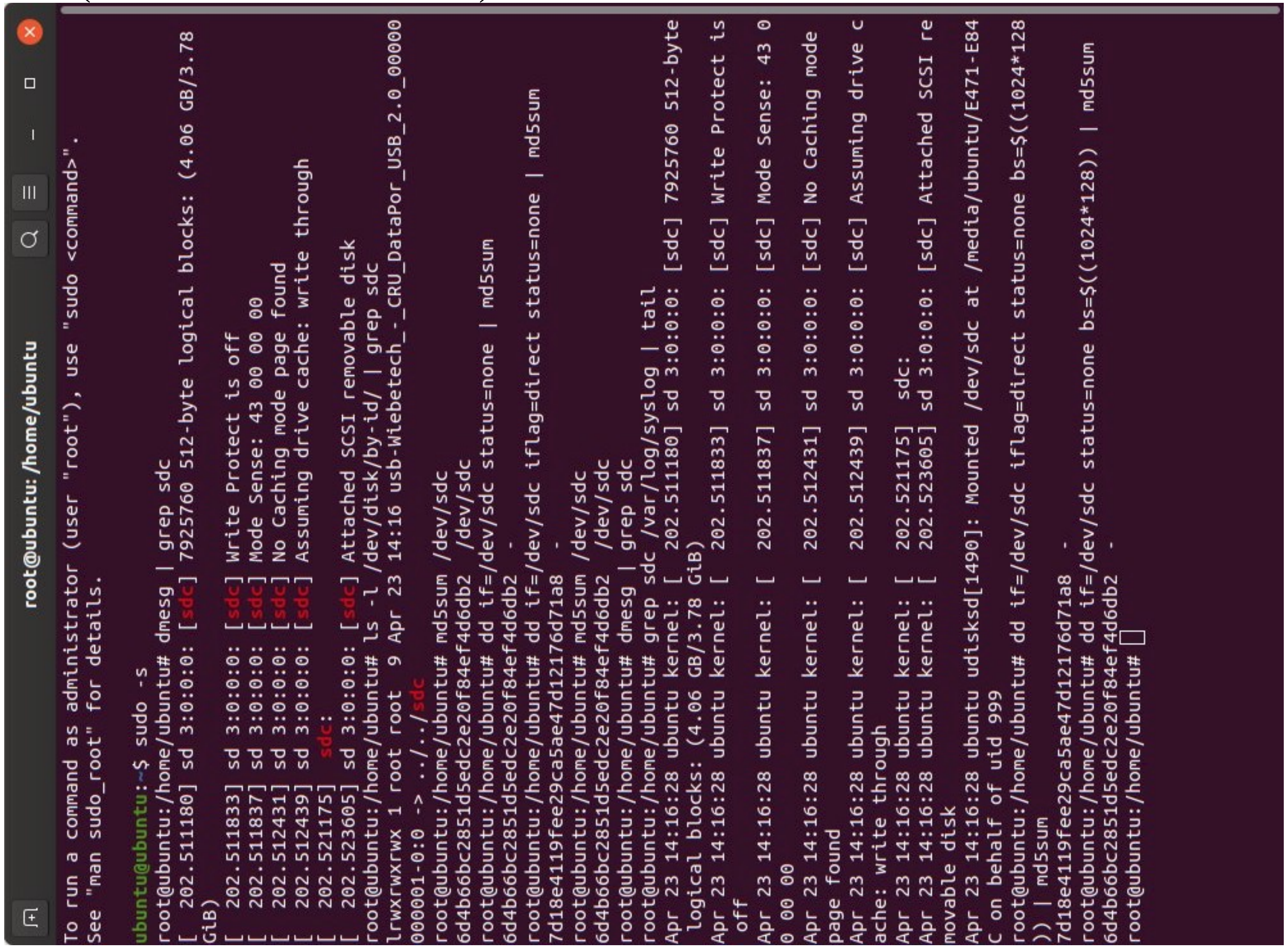
The “/dev/sda” device is not write protected, but during the second execution all writes are reported as blocked.

This greatly affects the reliability of test results for hardware write blockers.

2.1. The “read-write-read-compare” approach does not fully account data modifications performed by a hardware write blocker itself, with no command from the host (like in the Tableau TD3 case mentioned

		before). Also, this approach does not account bugs that allow overwriting a sector with the same data (i.e., when a hash value of a given sector remains the same, but its data is overwritten with a write command). Although no such issues were observed in practice. 3. The usage of hardware write blockers can introduce anomalies into acquired data, while source data (i.e., located on a source storage device) remains intact. CRU WiebeTech USB (2.0) WriteBlocker is a hardware write blocker designed for USB flash drives and other USB storage devices. It reports an attached storage device as writable and silently discards all write commands (no write error is reported to a host). An image created using this hardware write blocker can include sectors with data that differs from located on a storage device if the following conditions are met: - a Linux distribution is used to acquire the storage device; - this storage device contains a file system supported by the Linux distribution (e.g., FAT32); - this file system starts at LBA of 0 (no partition table is used); - this file system is mounted in the “read-write” mode before the acquisition (e.g., the automount feature is enabled); - the disk imaging tool does not use direct I/O requests. Under these conditions, the operating system serves read requests through its cache. Since the drive is reported as writable, and the file system is mounted in the “read-write” mode, and write errors are suppressed, write attempts successfully modify the cache but do not reach the storage device. Only read requests marked as direct bypass the cache and give data from the drive (but many disk imaging tools do not mark their I/O requests as direct: e.g., the dd tool does not use them by default). This means that modifications (like timestamp updates) made by the operating system touch the cache but not the storage device. Attempts to read data from the storage device are likely to produce data from the

cache (which contains modified sectors). A screenshot to demonstrate the issue is below:



```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ubuntu:~$ sudo -s
root@ubuntu:/home/ubuntu# dmesg | grep sdc
[  202.511180] sd 3:0:0:0: [sdc] 7925760 512-byte logical blocks: (4.06 GB/3.78
GiB)
[  202.511833] sd 3:0:0:0: [sdc] Write Protect is off
[  202.511837] sd 3:0:0:0: [sdc] Mode Sense: 43 00 00 00
[  202.512431] sd 3:0:0:0: [sdc] No Caching mode page found
[  202.512439] sd 3:0:0:0: [sdc] Assuming drive cache: write through
[  202.521175]  sdc:
[  202.523605] sd 3:0:0:0: [sdc] Attached SCSI removable disk
root@ubuntu:/home/ubuntu# ls -l /dev/disk/by-id/ | grep sdc
lrwxrwxrwx 1 root root  9 Apr 23 14:16 usb-Wiebetech_-_CRU_DataPor_USB_2.0_00000
000001-0:0 -> ../../sdc
root@ubuntu:/home/ubuntu# md5sum /dev/sdc
6d4b66bc2851d5edc2e20f84ef4d6db2  /dev/sdc
root@ubuntu:/home/ubuntu# dd if=/dev/sdc status=none | md5sum
6d4b66bc2851d5edc2e20f84ef4d6db2  -
root@ubuntu:/home/ubuntu# dd if=/dev/sdc iflag=direct status=none | md5sum
7d18e4119fee29ca5ae47d12176d71a8  -
root@ubuntu:/home/ubuntu# md5sum /dev/sdc
6d4b66bc2851d5edc2e20f84ef4d6db2  /dev/sdc
root@ubuntu:/home/ubuntu# dmesg | grep sdc
root@ubuntu:/home/ubuntu# grep sdc /var/log/syslog | tail
Apr 23 14:16:28 ubuntu kernel: [  202.511180] sd 3:0:0:0: [sdc] 7925760 512-byte
 logical blocks: (4.06 GB/3.78 GiB)
Apr 23 14:16:28 ubuntu kernel: [  202.511833] sd 3:0:0:0: [sdc] Write Protect is
 off
Apr 23 14:16:28 ubuntu kernel: [  202.511837] sd 3:0:0:0: [sdc] Mode Sense: 43 0
0 00 00
Apr 23 14:16:28 ubuntu kernel: [  202.512431] sd 3:0:0:0: [sdc] No Caching mode
page found
Apr 23 14:16:28 ubuntu kernel: [  202.512439] sd 3:0:0:0: [sdc] Assuming drive c
ache: write through
Apr 23 14:16:28 ubuntu kernel: [  202.521175]  sdc:
Apr 23 14:16:28 ubuntu kernel: [  202.523605] sd 3:0:0:0: [sdc] Attached SCSI re
movable disk
Apr 23 14:16:28 ubuntu udisksd[1490]: Mounted /dev/sdc at /media/ubuntu/E471-E84
C on behalf of uid 999
root@ubuntu:/home/ubuntu# dd if=/dev/sdc iflag=direct status=none bs=$((1024*128
)) | md5sum
7d18e4119fee29ca5ae47d12176d71a8  -
root@ubuntu:/home/ubuntu# dd if=/dev/sdc status=none bs=$((1024*128)) | md5sum
6d4b66bc2851d5edc2e20f84ef4d6db2  -
root@ubuntu:/home/ubuntu# 
```

The same USB flash drive produces different hash values depending on whether I/O operations are direct (the “iflag=direct” argument is given) or not.

51	**KEY TAKEAWAY #4.7:** Extensive	It is not clear which anomalies are considered as minor. Since the root cause of many anomalies is often

	tool testing of over 250 widely used digital forensic tools showed that most tools can perform their intended functions with only minor anomalies.	not revealed, it is unclear how to differentiate between minor and major ones. For example, if a disk imaging tool modifies some data on a source storage device, this modification can be considered as minor when only several sectors are affected. However, further tests can demonstrate that the same issue can modify thousands of sectors (depending on the file system state found on the storage device). A more specific example is the ASR Data SMART live CD (version 2011-01). According to the report (https://www.dhs.gov/sites/default/files/publications/508_Test%20Report_NIST_Digital%20Data_ASR_Data_SMART_September%202012.pdf): The execution environment, the SMART Linux live CD version 2011-01, not the tool, modified the source drive in test cases DA-02-F12, DA-02-F32, and DA-06-ATA28. The source drive, 01-IDE, contained an NTFS and several other file systems. In each case 88 sectors belonging to the NTFS file system journal were changed. Since the execution environment's changes were limited to the NTFS partition, the accuracy of the DA-02-F12 and DA-02-F32 acquisitions (acquisitions of the drive's FAT 12 and FAT 32 partitions) were not affected. However, in DA-06-ATA28 this resulted in 88 sectors differing between the image file created by the tool and the original unaltered source. When the test cases were rerun with the source attached via hardware write block (DA-02-F12-WB, DA-02-F32-WB and DA-06-ATA28-WB), the tests completed without anomaly. It should be noted that in testing SMART, other drives that contained NTFS file systems were imaged but were not modified by the SMART Linux environment. This behavior of SMART Linux changing the source was only seen with the NTFS file system on drive 01-IDE. According to another research (https://dfir.ru/2018/07/25/a-live-forensic-distribution-writing-to-a-suspect-drive/), this issue was caused by the live environment mounting NTFS file systems in the "read-write" mode during the boot. This results in the $LogFile journal being wiped if the file system was not unmounted properly before the test. So, the issue affects more than just 88 sectors on an IDE drive. And since the root cause was not analyzed, the same issue existing in other live distributions tested (e.g., SUMURI PALADIN 6.09) was not discovered.

55	A real-world data set also has disadvantages: […]	Privacy concerns and copyright issues can be mentioned as disadvantages of real-world data. Even simulated data can include unwanted personal information. And even simple file system images can include copyrighted material (like boot code embedded into the NTFS file system).

Chapter 5

Page	Text	Comments
56	Currently, digital forensics labs are each testing the same tools causing redundant work.	Again, it is essential to test the same tools in different environments. Even the same version number of a tool does not guarantee that this tool does not produce different output for the same input data. *More information about this topic is provided above (see page 9 of this document).*
56	Better analysis of how digital evidence is used and whether there have been incorrect or misleading conclusions. Having this information centrally collected would benefit the field.	It should be noted that many issues with digital forensic tools are discovered outside of the NIST tool testing programs and projects. Such issues are often reported directly to vendors. In many cases, especially with proprietary software, these reports and related change log entries are never made available to the public (i.e., bugs are fixed quietly). For example, one popular mobile forensic tool always reported the inode changed timestamp as the file created timestamp when parsing the F2FS file system (this raised questions about the origin of a photo in one case). This issue was reported to the vendor, the vendor fixed it (and before confirming the bug, the vendor wrongly insisted that all timestamps are reported correctly). However, no change log entry (and no announcement in any form) that somehow describes the fix is available. (The vendor and the tool are not named here for legal reasons.) A good idea is to create a public issue tracker for bugs discovered in digital forensic tools, so all practitioners could report an issue in any digital forensic tool. The issue tracker can be useful when dealing with vendors who do not accept bug reports for some reason.
56	While developing this report, we encountered many areas that need further research and improved	There is another important area: - Better understanding of how forensic examiners rely on tools.

	processes, including: *[…]*	Forensic examiners rely too much on tools. But can forensic examiners effectively uncover previously unknown errors in tools they use? Especially, when tools produce expected (but incomplete or even incorrect) results. How many tool errors are mistakenly documented as proper behavior?