The dependence of the performance of imaging tasks on the forensic imager interface type

F B Tebueva, M G Ogur, M V Zubkov, M G Moisov
North-Caucasus Federal University, Stavropol, Pushkin str., 1, 355017, Russia
E-mail mogur@ncfu.ru

Abstract. Forensic research in the investigation of information security incidents, the production of forensic examinations and many other areas of activity related to computer forensics require the maximum possible preservation of the integrity of the investigated data. To do this, write blockers are used - programs or devices that do not allow writing anything to the drive under investigation. The need to use such means comes both from the requirements of procedural legislation (for example, the Criminal Procedure Code of the Russian Federation), and from various recommendations of a methodological and other nature, as well as from standards (for example, STO BR IBBS-1.3-2016). Some aspects of the functioning of write blockers will be discussed in this article. And on the example of the criminalistic lock the question of influence of the used interface for data transmission in tasks of creation of images of data carriers will be considered.

Keywords: forensic; forensic imager, interface, performance, hardware bridge, software bridge.

1. Introduction
In the modern world the special relevance is acquired by a problem of commission of crimes with use of the computer equipment. As every year information volume grows in the world, criminalists at investigation of such crimes are forced to work with huge data array. As one of the tools used for the analysis of information is the record lock, important aspect in work with this device is the type of a host interface and according to its capacity. In this regard the purpose of this article is to show influence of capacity of various interfaces on the speed of creation of images of data carriers [2,3,4].

2. Basics of forensic imaging
Many experts in computer criminalistics and lawyers share opinion that hardware bridges of record are more reliable, than program bridges of record; this judgment still can be found, directly or indirectly, in various publications [1,2].

2.1. Principles of functioning
Hardware bridges – all hardware bridges of record can be divided into two groups how they process the teams received from a host:

- working at base of the white list;
- working at base of the black list.

The hardware bridge works at base of the white list when it bridges any team to the store if it is not
included in the list of the famous safe teams (which are not making changes to the data stored on the store). In this operating mode the lock of record will block all unknown teams, including specific to the producer (for example, for performing low-level diagnostics of the store) and new (the records which are not realized in the built-in program of the lock yet). Such lock of record can block the new standardized safe teams, interpreting them as unknown.

The hardware bridge works at base of the black list when it bridges the teams included in the list of the famous unsafe teams (which are making changes to the data stored on the store or carrying out other dangerous actions), and allows passing to the store of any other teams. In such operating mode the lock of record will allow unknown (specific to the producer or new standardized) unsafe teams to the store.

Besides, all hardware bridges of record can be divided into two other groups depending on details of their realization:

- working as the translator of teams;
- working as the modules providing access to the block device.

The hardware lock of record works as the translator of teams when it just broadcasts the allowed teams received from the interface source by their repetition in the interface recipient. For example, the simple lock of SATA-to-USB record can receive SCSI commands from the USB interface (using a set of the teams "SCSI transparent command set" for the class "mass storage"), and then to make inquiry for each allowed SCSI command to the SATA controller through AHCI, redirecting any answers back to a host under the SCSI protocol.

The hardware lock works as the module providing access to the block device when it contains the full operating system of general purpose, the connected store is defined as the block device in this operating system, and access to reading from this block device is divided with a host via the special driver. Such lock of record with USB connection to a host will define the connected store as the block device, and then will use the USB gadget for emulation of the USB store with use of the specified block device as data source for the emulated store. In such configuration the hardware lock of record can carry out direct broadcast of the teams received from a host to the store, it broadcasts the teams received from a host in the internal inquiries used for data reading from the block device source. Thus, the multiple teams of reading received from a host can be connected in one request for reading leading to sending to the connected store of one team of reading. Besides, the built-in program of the lock of record can carry out anticipatory reading (read-ahead) in a cache, it leads to the fact that one team of reading received from a host can not lead to immediate sending the corresponding team of reading to the connected store as the requested data were already read and added to a cache by the built-in program.

Besides, hardware bridges of record can provide special functions for some typical and atypical applications:

- work in the reading record mode;
- permission of commands to write, saving the modified data on other store;
- representation of the store to a host with a mark of write protection (it adds one more level of protection as it is expected that the operating system will not write on the store with a mark of write protection);
- concealment of write errors;
- providing access to hidden by means of HPA or DCO areas of data;
- permission of some unsafe teams used for opening of access to the hidden areas of data (removal of DCO or permanent removal of HPA);
- transparent proofreading of data from the bad (damaged) sectors of the store, transparent work with faulty stores.

Program bridges – details of realization of program bridges of record depend on the used operating system. In the operating systems working in the real mode like DOS, program bridges of record intercept the interruption of BIOS 0x13 used for reading and data recording of a disk, filtering requests for record and causing the initial handler of interruption for requests for reading. Modern operating systems like Windows and GNU/Linux use drivers of direct access for interaction with stores, interruption of BIOS
0x13 are used by the loader for reading a kernel and other data (like drivers of direct access) only at an early stage of loading. Thus, there are set of ways for realization of functionality of blocking of record, for example:

- the driver working in the mode "only reading" for a certain class of stores (PATA, SATA, SCSI, USB, etc.);
- the program (driver) filtering requests for record for their ways to the driver of a certain class of stores of lower level;
- the driver providing the block device in the mode "only reading" for the chosen store or the section with parallel existence in the operating system of the block device in the reading record mode for the same store or the section (it is supposed that use of various programs will happen in relation to the block device in the mode "only reading").

Depending on realization, program bridges of record can analyze or block the following types of inquiries:

- reading, record, dumping of a cache and other inquiries in the unified format specific to the operating system (which is not based on the protocol of interaction with the store at a low level);
- inquiries in the format based on the protocol used for interaction with the store (for example, SCSI) or which directly realizes this protocol.

Besides, the program lock of record can present the store to the operating system with a mark "only reading".

In modern operating systems the following realization of blocking of record was met:

2.2. Windows:

- installation of the filtering driver for packages of I/O-inquiries which are used for transfer of SCSI commands low-level port driver of the store (the operating system is used the SCSI protocol for interaction about port drivers of stores, packages of inquiries are broadcast if it is necessary, in the protocol used by the concrete equipment).

2.3. Linux:

- installation of the device - "loop" in the mode "only reading" for the block device of the store (or undressed), in this situation the driver of the device - "loop" filters the requests for record going to the main block device (the kernel uses own structure for the description of requests to the store on reading, record, dumping of a cache, etc.), use of the device - "loop" for access to data is supposed;
- a kernel patching for the purpose of filtering out of the requests for record going to the block device in the mode "only reading".

During the work with the inquiries created on the basis of the native protocol (like SCSI in Windows), the program lock of record can work with black and white lists as it was shown earlier.

In general, program bridges of record have to intercept inquiries or in one point (for example, before inquiry transfer to one of many drivers of stores), or in all places at once (for example, in all drivers of stores).

It should be noted that the program lock of record cannot block all possible ways of transfer of the unsafe command to the store. For example, the kernel can provide the interface for sending "crude" inquiries to the store (like the SG_IO interface in Linux), or the architecture of drivers of stores can allow any driver to send a request bypassing the filtering driver (as in Windows), or the low-level driver can independently send unsafe teams, or any program can just switch-off the record lock. And though it is possible to take some measures against such shortcomings, the idea is simple - there is always a way bypassing the program lock of record.

2.4. Program quasibridges of record

There is an opportunity to create such operating system which will not send unsafe teams to the connected stores to time and after loading, except situations when the user obviously starts the program sending unsafe requests. In that case there is no component blocking record, but there are also no teams
which are subject to blocking (and such operating systems often carry to containing the record blocking mechanism therefore in this section and designation "quasi" is used)

Unfortunately, some products are described how the records containing the program lock, but in fact in them are not present such lock, and depending on various circumstances the similar product can send to the connected store unsafe teams.

2.5. **Comparison of hardware bridges of record with program**

The following important differences between hardware and program bridges of record can be noted:

- Hardware bridges of record do a gap between a host and the store, as a result unsafe commands from a host are subject to blocking regardless of their origin. Program bridges of record can be bypassed by the malicious application as it was shown earlier.

- Program bridges of record for drivers of direct access are inactive during an early stage of loading: blocking of record is absent when the loader uses interruption BIOS 0x13 or EFI services for loading of a kernel and other components of the modern operating system. At the same time hardware bridges of record before completion of their initialization do not process any teams.

There is an opinion that program bridges of record cannot be reliable because depend on the fragile program environment: for example, updating of the operating system, either updating of the driver, or a mistake (in hardware or the software) can interfere with the record blocking mechanism; hardware bridges of record, on the other hand, are considered as reliable because contain the stable, well tested hardware and the built-in software.

3. **Formation of tasks of an influence research a host interfaces on final performance**

The following actions will be made for achievement of an objective:

- Copying of the test hard drive SATA III connected via the record lock via USB 3.0 interface.
- Copying of the test hard drive SATA II connected directly via the USB 3.0 interface.
- Copying of the test hard drive SATA III of the disk connected directly via the SATA interface.
- Copying of the test hard drive SATA II of the disk connected directly via the SATA interface.
- Comparison with SATA SSD

As the criminalistic lock of record Tableau T3iu which connection is carried out via the USB interface is used. The used software for copying is the X-Ways forensics complex [5,6].

3.1. **Practical research**

In the first case the copying of the classical hard drive on 1tb, connected via the record lock is considered.

![Copying of sectors... Number 4024320](image)

Figure 1. Copying of SATA III of the hard drive via the USB 3.0 interface.

Copying is made on other hard drive connected directly via the SATA III interface. In this situation speed of performance of copying rests against the maximum capacity of the USB 3.0 interface that when copying enough volume data carriers can slow down implementation of the analysis of collected data. At the exit we receive speed in 6.4 GB/min. that will allow to make an image of this disk for 2.4 h.
Figure 2. Copying of SATA II of the hard drive via the USB 3.0 interface.

As well as in the first case the record is made on the same disk, but as copied the disk which does not support the SATA III mode was used. At the exit we receive speed in 4.3 GB/min. that for 33% more slowly than a disk from the first experience. Therefore, time spent for copying will be 3.6 h.

In the third case the hard drive by volume on 1tb, but connected directly via the SATA II interface owing to the fact that it does not support the SATA III mode was used [9].

Figure 3. Copying of the hard drive connected directly via the SATA II interface.

At this investigation phase it is possible to tell that direct connection to the SATA II interface does not concede on copying speed to a disk with support of SATA III which was connected via the lock to the USB 3.0 interface. Concerning the first experience the speed is 0.2 GB/min. less that at the level of an error of the taken measurements. The image of a disk will be created in 2.5 hours.

Figure 4. Copying of the hard drive connected directly via the SATA III interface.

At connection of a disk to the SATA III interface the natural growth of speed of copying as capacity grew from 3 Gbps to 6 Gbps is observed. But despite the increased capacity of high speeds it is impossible to achieve owing to the structure of the hard drive and existence in it of mobile parts. Speed is 7.1 GB/min. that is 10% faster than a disk from the first experience. The image will be created in 2.2 hours.
In the last experience the copying of a SSD disk with the SATA III interface is considered. Speed of copying is much higher than classical stores with mobile parts that allows to achieve such result. Speed of creation of an image is 9.7 GB/min. that quicker:

1. About 34% faster than the first disk;
2. 51.5% faster than the second disk;
3. 36% faster than the third disk;
4. 26.8% faster than the fourth disk.

It is not possible to compare time of creation of an image as the volume of the considered SSD disk is about 4.2 times less [6,7,8].

4. Conclusion

During this research it was established that in practice use of the lock with the interface of connection of USB 3.0 can affect considerably the speed of implementation of the copy of a disk that in turn can increase time expenditure by implementation of the analysis of collected data [10,11]. Thus, it is expedient to use an analog of the considered lock which will have support of the SATA interface thanks to what decrease in time for the analysis of data as creation of an image for further work with it will take less time owing to the increased speeds will become possible [12].

Also, during the research, the following was revealed:

- the Disk SATA III connected via the lock to the USB 3.0 interface more slowly similar, but connected directly to SATA III for 10%
- the Disk SATA II connected via the lock to the USB 3.0 interface more slowly similar, but connected directly to SATA II for 36%

References

[1] Gary C. Kessler, Gregory H. Carlton 2014 A Study of Forensic Imaging in the Absence of Write-Blockers (Journal of Digital Forensics, Security and Law: Vol. 9 : No. 3 , Article 4.)
[2] Carrier, B. 2019 Open Source Digital Forensics Tools,
[3] James, J.I.; Gladyshev, P. 2015 Automated inference of past action instances in digital investigations Int. J. Inf. Secur. 2015, 14) pp 249-261. [CrossRef]
[4] Sitompul, O.S.; Handoko, A.; Rahmat, R.F. 2018 File Reconstruction in Digital Forensic (TELKOMNIKA Indones. J. Electr. Eng. 2018, 16) pp 776-794. [CrossRef]
[5] Kalber, S.; Dewald, A.; Freiling, F.C. 2013 Forensic application-fingerprinting based on file system metadata (Proceedings of the Seventh International Conference on IT Security Incident Management and IT Forensics, Nuremberg, Germany, 12-14 March 2013) pp 98-112.
[6] Bang, J.; Yoo, B.; Kim, J.; Lee, S. 2009 Analysis of time information for digital investigation (Proceedings of the Fifth International Joint Conference on INC, IMS and IDC, Seoul, Korea, 25-27 August 2009) pp 1858-1864.
[7] Chabot, Y.; Bertaux, A.; Nicolle, C.; Kechadi, M.T. 2014 A complete formalized knowledge representation model for advanced digital forensics timeline analysis (Digit. Investig. 2014, 11)
pp 95-105. [CrossRef]
[8] Hargreaves, C.; Patterson, J. 2012 An automated timeline reconstruction approach for digital forensic investigations (Digit. Investig. 2012, 9) pp 69-79. [CrossRef]
[9] Brady, O.; Overill, R. 2015 DESO: Addressing volume and variety in large scale criminal cases (Digit. Investig. 2015, 88) pp 72-825. [CrossRef]
[10] Brady, O.; Overill, R.; Keppens, J. 2014 Addressing the increasing volume and variety of digital evidence using an ontology (Proceedings of the 2014 IEEE Joint Intelligence and Security Informatics Conference, Hague, The Netherlands, 24-26 September 2014) pp 176-183.
[11] Debinski, M.; Breitinger, F.; Mohan, P. 2019 Timeline2GUI: A Log2timeline CSV parser and training scenarios (Digit. Investig. 2019, 28) pp 34-43. [CrossRef]
[12] Soltani, S.; Seno, S.A.H.; Yazdi, H.S. 2019 Event reconstruction using temporal pattern of file system modification (IET Inf. Secur. 2019, 13) pp 201-212. [CrossRef]