Proceedings of the Global Engineering, Science and Technology Conference 2012
28-29 December 2012, Dhaka, Bangladesh

Secure Data Shredder

Vikram Bahl, David Leong, Guo Jiayan, Jonathan Siang and Tay Mei Lan

In our current information explosion era, there are massive amounts of confidential data belonging to individuals, companies and governments residing on data storage devices. It is important to be able to securely and irrevocably destroy this data before these devices are discarded or reused, so that it cannot be extracted and exploited by malicious parties. There are open source solutions available to destroy data permanently. However, they are limited in their ability to provide a vast choice of data destruction algorithms and have the constraint of supporting a single operating system. This has inevitably led to a small group of pertinent users. In addition, they are unable to destroy data on multiple storage devices at once, thus relatively slowing down performance. In order to overcome these pitfalls, we have developed an open source cross platform data shredding software that could securely, rapidly and permanently destroy information stored on any prevalent data storage device used in computers and embedded systems. To ensure speedy processing, our application has provided the functionality to wipe multiple storage devices concurrently. We have ensured that it complies with the IEEE T10/T13 specifications and operates with no limitations to the capacity of the storage media connected to the computer system. We have implemented several techniques defined in various proprietary standards to handle data remanence. We have also included a selection of military grade data destruction procedures recommended by renowned information security specialists. In order to serve users with various operating systems, our application is interoperable in Windows, Mac OS X and Linux environment. In comparison with other popular open source solutions, our software is found to be secure, reliable, coherent and has the ability to gratify large varieties of home and corporate users.

Field of Research: Computer Software and Application

1. Introduction

The proliferation of data storage devices per user for personal and corporate use has resulted in copious amounts of personal and confidential information being digitally stored across a number of storage devices. When this data is to be deleted, users usually do so by employing commonly used methods depending on their operating system, like pressing the *Delete* key on the keyboard and proceeding to remove the item from the *Recycle Bin* or *Trash* folder. However, performing such an operation on the data only removes it from the entries in the file system journals. While the user may not be able to locate and access the data, the content still resides in locations on the hard disk platter and is easily retrievable by commonly available utilities (Wiles and Reyes, 2007). Deleted data does not disappear from the hard disk abruptly until the user starts writing data over it. Reformatting of hard drives does not eradicate data from the computers either.

Vikram Bahl, Republic Polytechnic, Singapore.Email vikram_bahl@rp.edu.sg
David Leong, Republic Polytechnic, Singapore. Email david_leong@rp.edu.sg
Guo Jiayan, Republic Polytechnic, Singapore. Email guo_jiayan@rp.edu.sg
Jonathan Siang, Republic Polytechnic, Singapore. Email jonathan_siang@rp.edu.sg
Tay Mei Lan, Republic Polytechnic, Singapore. Email tay_mei_lan@rp.edu.sg

It merely updates and refreshes the *superblock* depending upon the file system format chosen upon reformatting (Deitel, 2004). The data cannot be located since its entry from the file system table has been erased but it is recoverable since it still resides on the disk media. Unskilled malicious hackers are able to retrieve files (even those deleted a long time ago) using off-the-shelf data recovery tools that are easily available on the Internet, for example, Recuva (Pirifrom, 2012) and TestDisk (CGSecurity, 2012), thus allowing malicious parties access to confidential information thought to be deleted by the user. In a day and age where data from personal photographs, financial documents and passwords to sensitive information relating to national security is digitally stored, it becomes imperative to ensure that deleted data is securely and permanently erased from the storage device.

In a set of guidelines published by the National Institute of Standards and Technology (NIST), users are advised to perform *secure erase* procedures, also known as *disk wiping* or *data shredding*, on old reusable hard disks in order to irrecoverably expunge digital data (NIST, 2006). We conducted a study on various data destruction algorithms and found various military grade procedures proposed by information security specialists (Gutmann, 1996; Gutmann, 2001). We also surveyed some open source solutions to data destruction and found them to be limited in the variety of destruction algorithms supported. They were also not cross-platform and supported only a limited range of storage media. We then proceeded to design a cross-platform, secure and comprehensive data destruction utility. The aim of this paper is to present the methodology and capability of the data shredding software developed by us. Our software is able to sanitize multiple hard disks and flash drives simultaneously. Further, our application allows the user to permanently overwrite data using predefined military grade data destruction algorithms. The ultimate goal of our software is to annihilate all previous data entries and partitions, so as to make it nearly impossible for a malicious user to recover any readable data that was previously stored on the storage device.

2. Literature Review

The most common method employed by users to delete files from their system is to press the *Delete* key and periodically empty the *Recycle Bin* or *Trash* folders. Users then have a false sense of security that the data has been permanently deleted from the system. However, the *Delete key* operation does not remove the data but only the pointer to that data. If the entry in the file allocation table is removed or the *inode* is removed from the *inode table* (Bach, 1986), the ‘deleted’ data remains on the storage media as unallocated space until it is overwritten. Some users may also choose to format their storage media. While this option may seem more secure and permanent as compared to pressing the *Delete* key, it does not completely erase the data from the hard disk. It only removes the reference to that data in the file system journal. Data storage can be compared to a book with a large index at the beginning. The index contains the chapter, topic and subject information which is a reference to the actual data contained in the book. This index is managed by a file system like NTFS or FAT. Upon formatting, the file system deletes this index and plants a fresh journal entry making sure that the user is unable to locate the data. The ‘deleted’ data still resides on the hard disk, but the operating system shows it as free space. The data can still be recovered by creating virtual pointers to ‘lost’ data. Other

methods to irrevocably destruct data include degaussing and physical destruction of the hard drive. Degaussing exposes the magnetic storage media to a strong magnetic field to scramble data stored on a magnetic tape/drive. The physical destruction of storage media involves melting, pulverization, burning and incineration of the disk (NIST, 2006). These two methods render the drive unusable, are expensive and pose a threat to personal safety and put the environment at risk. Software-based solutions for data destruction usually espouse overwriting existing data with specific bit patterns. The US Department of Defence proposed a 3 passes overwriting algorithm of zeroes, ones and a random bit pattern (Defense, 1995). Peter Gutmann had also proposed a 35 passes overwriting algorithm that may resolve the probable issues encountered in data remanence (Gutmann, 1996; Gutmann, 2001). These methods could render the old or existing data unreadable.

3. Methodology

3.1 Data Destruction Algorithms

Our application allows the user to choose from 10 wipe methods for data destruction. While each of the methods will completely destroy the data, the user has the option of using multiple methods to wipe the disk. The details of the data destruction methods are given in *Table 1*. Data destruction implies that the storage device from which data is to be wiped is overwritten with a certain bit pattern a number of times. Overwriting the storage device once with a bit pattern is called *1 pass*. Usually, a *verification pass* is done to ensure that correct data is overwritten. *Table 1* gives an overview of the algorithms implemented. Further, our software permits the user to erase the disk with predefined patterns: 0x00, 0xFF and a random pattern. Moreover, it also allows the user to define their own bit pattern supporting up to 30 ASCII characters.

Table 1: Overview of data destruction algorithms implemented

Data Destruction Algorithm	Overwriting Pass	Verification Pass
• British HMG IS5-Baseline	Pass 1: 0x00	1
• British HMG IS5-Enhanced	Pass 1: 0x00 Pass 2: 0xFF Pass 3: *Random*	1(on Pass 3)
• Russian GOST P50739-95	Pass 1: 0x00 Pass 2 : *Random*	0
• US Standard DOD 5220.22-M	Pass 1: 0x00 Pass 2: 0x01 Pass 3 : *Random*	3
• Canadian RCMP DSX	Pass 1: 0x00 Pass 2: 0x01 Pass 3 : *Random*	3

• Canadian RCMP TSSIT OPS-II	Pass 1, 3, 5: 0x00 Pass 2, 4, 6: 0x01 Pass 7: *Random*	1 (on Pass 7)
• German VSITR	Pass 1, 3, 5: 0x00 Pass 2, 4, 6: 0x01 Pass 7: *Random*	0
• Bruce Schneier's Algorithm	Pass 1: 0x01 Pass 2: 0x00 Pass 3-7: *Random*	0

Apart from the algorithms described in *Table 1*, the user is also given the option to execute Peter Gutmann's proposed algorithm that prescribes 35 overwrite passes (Gutmann, 1996). In this scheme, passes 1-4 and 32-35 are random bit patterns overwritten on the hard drive. Passes 5-31 are designed by taking into consideration specific magnetic media encoding schemes. There is an underlying sequence to the bit patterns for sequential passes. The bit pattern for Pass 6 is obtained by a bitwise left shift rotate from Pass 5. This means that each bit in Pass 5 (**0101**01010101010101010101) is shifted one place to the left to obtain Pass 6 (**101**01010101010101010101**0**). This bitwise left shift is equivalent to doubling the value of an integer number. Similarly, bitwise right shift is corresponding to halving an integer number. Passes 7-9, 26-28 and 29-31 are bitwise right shift rotations. The hexadecimal value for Passes 10-25 is to be increased by one for every pass (i.e. from 0x00 to 0xFF). *Table 2* gives an overview of the bit patterns used in Gutmann's algorithm.

Table 2: Bit patterns in Peter Gutmann's algorithm

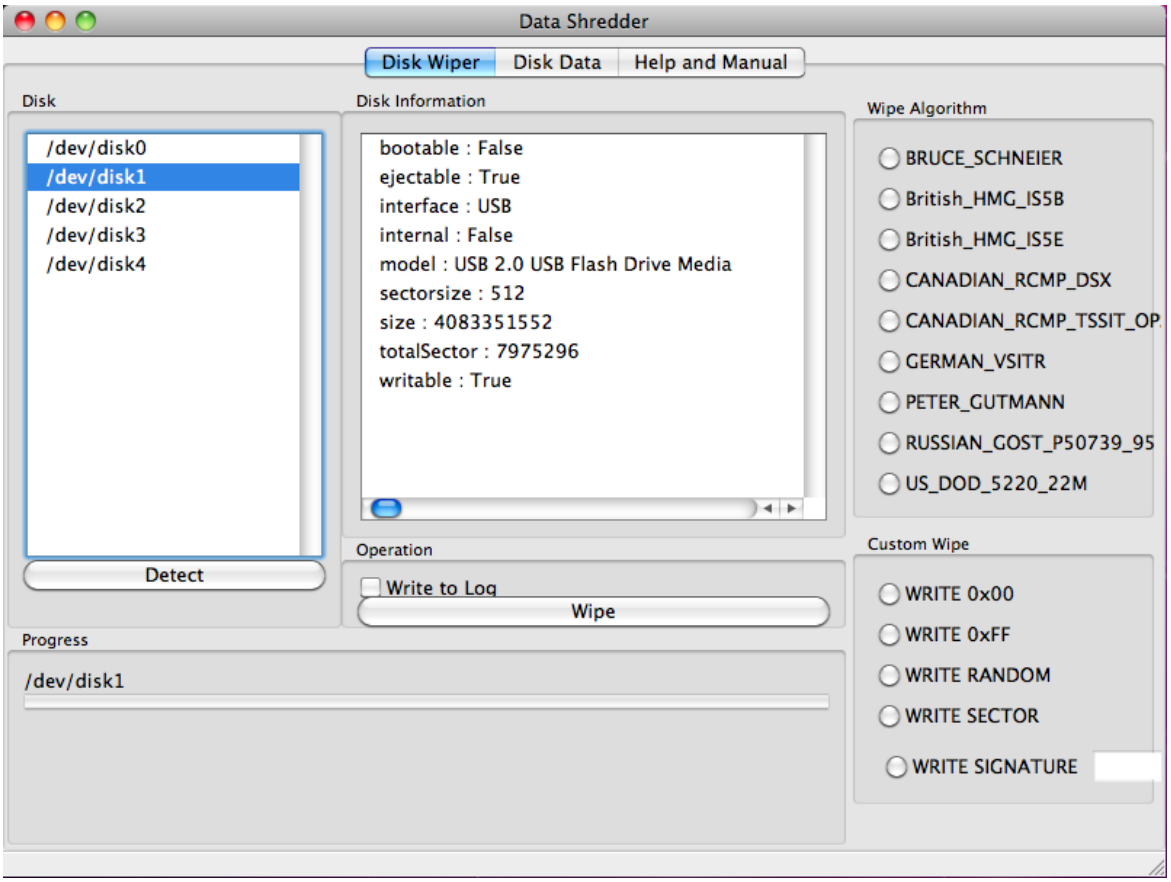
Pass	Overwritten bit pattern	
	Binary Notation	**Hex Notation**
1-4	*Random*	*Random*
5	**0101**0101 **0101**0101 01010101	55 55 55
6	10101010 10101010 10101010	AA AA AA
7-9	**100**10010 01001001 00100**100** **010**01001 00100100 10010010 00**100**100 10010010 01001001 *(Each bit from Pass 7 to 9 shifted one place to the right)*	92 49 24 49 24 92 24 92 49
10	**0000**0000 00000000 00000000	**0**0 00 00
11	**0001**0001 00010001 00010001	**1**1 11 11
12	**0010**0010 00100010 00100010	**2**2 22 22
13-25	*Bit patterns from 10 to 25 increase by 0x01 (00 00 00 – FF FF FF)*	33 33 33 – FF FF FF
26-28	*Same as Pass 7-9*	92 49 24 - 24 92 49
29-31	**011**01101 10110110 11011011 **101**10110 11011011 01101101 1**101**1011 01101101 10110110 *(Each bit from Pass 29 to 31 is shifted one place to the right: bitwise right rotate)*	6D B6 DB B6 DB 6D DB 6D B6
32-35	*Random*	*Random*

3.2 Software Features and Supported Hardware

3.2.1 GUI

The main UI screen of our software is shown in *Figure 1*. The left pane shows a list of storage devices mounted onto the operating system. The right pane shows a list of available data destruction methods to be selected by user. In accordance with standard UI design guidelines (Schneiderman, 2003), status and progress bars have been included. Our application permits the user to generate reports after each wipe operation to ensure security auditing as well as monitoring the performance of disk I/O. Further, a disk viewer utility which allows the user to view the data content in the disk sectors after each wipe pass is provided. This gives the user a clear visual depiction of the data that has been overwritten. Detection and recognition of storage devices, as well as the presentation of device information are emphasized in the subsequent sections.

Figure 1: Application user interface depicting a variety of wiping algorithms



3.2.2 Extracting Device Information

Our application automatically scans through the computer system to identify the type of operating system used when it is launched. The user is able to select individual storage devices to view critical device information like the hard drive model, serial number, firmware revision, capacity, total number of cylinders, heads, sectors, sector size, total number of sectors available, maximum value of the Logical Block Addressing (LBA) and many more. For the Windows platform, the Windows Management Instrumentation (WMI) library is used to extract system information from the storage device. WMI is Microsoft's implementation of the Web-Based Enterprise Management (WBEM) and Common Information Model (CIM) standards. Similarly, for the Linux platform, the commands *fdisk* and *lshw* are used to extract detailed information of disk and hardware configuration in the computer system.

3.2.3 Generating Pseudo-Random Numbers

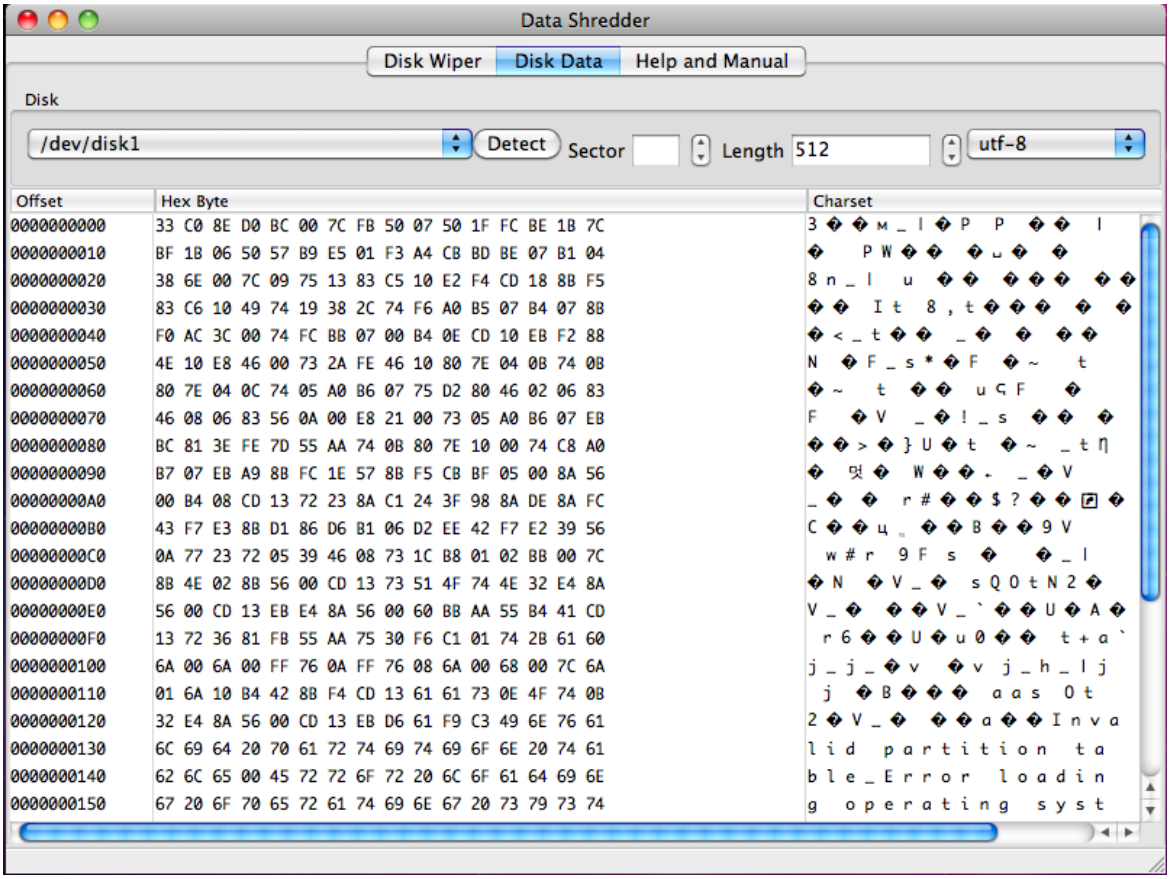
Many of the data destruction algorithms described in *Section 3.1* use pseudo-random numbers to overwrite data. We have utilized library calls which use the *Mersenne twister sequence* (Makoto & Takuji, 1998) to generate pseudo-random numbers. While this method is deterministic, it contains a few additional probability distributions commonly used in scientific research, as well as a couple of

convenience functions to generate random data. A three pass wipe of alternating zeroes and random numbers is sufficient for home users. However, military and corporate data typically require at least 7 or more passes in order to elude any possibility of data remanence (Gutmann, 1996; Gutmann, 2001).

3.2.4 Viewing Data Sectors

Our application provides a built-in *Disk Viewer* utility for data read-back verifications to the user. This permits the user to view the content of any sector before and after data wiping. For instance, the content of Volume Boot Record (VBR) from a 512 MB USB flash drive, as shown in *Figure 2*, can be viewed by UTF-8 or ASCII encoding. However for security reasons, functionality to edit data at specific byte position was not provided.

Figure 2: Software disk viewing utility



3.2.5 Software Development Process

Our application was developed in the Python programming language. The user interface was designed using wxPython, a set of Python bindings to the wxWidgets library which is a cross-platform C++ GUI application framework. The advantage of using wxPython is that it allows the user interface to be developed using the same controls and themes as the system on which it is deployed. It can be rapidly developed on one operating system and ported to another with minor changes

(Precord, 2010).The Model-View-Controller architecture (MVC) was used to design the application. The software development lifecycle (SDLC) was divided into several software modules each of which was developed concurrently and finally integrated.

3.2.6 Supported Hardware and Operating Systems

Table 3 lists the storage type, hardware and operating systems supported by our software.

Table 3: Hardware, software and device support

Storage Type Support	Up to 8 PATA/SATA/SCSI/SAS/USB hard drives, Solid State Drives, NVRAM, CD/DVD drive, USB flash media.
Software Support	Windows 8/7/Vista/XP, Red Hat Enterprise Linux 6.2, MAC OS X.
Device Support	Laptop, Desktop PC, Workstation, Servers.
Minimum Sector/Block Size	512 bytes
Bootable Devices	Flash drive, CD/DVD drive.

4. Experimental Results and Discussion

4.1 Managing Concurrency In Data Wiping

Due to slow disk I/O incurred on individual storage device during a wipe process, the wipe operations are implemented using a concurrency approach. Multiple threads are deployed by us to this effect. Threading is a technique to decouple tasks that are not sequentially dependent. The threading module is engineered to fork multithreads for the application to handle wipe operations on multiple storage devices concurrently, thus improving performance.

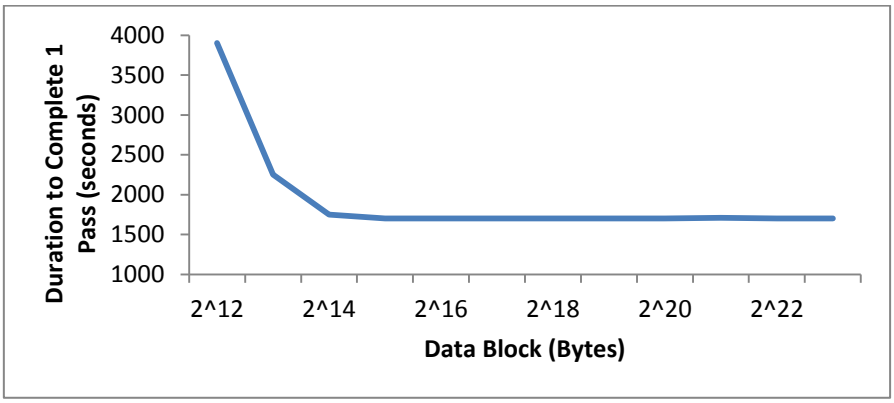
4.2 Performance Testing and Optimization

Upon data shredding, the slack space in the hard disk was verified for data remanence using the AccessData's Forensic Toolkit FTK 4 software and no trace of any old data was found. Furthermore, we computed the SHA-1 hash value of the disk image and it coincided with the last wipe pattern performed by our software. These validation tests have affirmed that the data destruction process we had performed is secure.

Performance testing was carried out on a hard drive using a single overwrite pass. Upon investigation, the performance bottleneck was identified to be at the disk I/O. We concluded that wiping data sector-by-sector was too sluggish. We then proceeded to improve the I/O performance by implementing *block buffer optimization* technique. The wipe performance of our software improves drastically when the buffer size escalates. However, buffer size larger than 32 KB does not yield further improvement to the overall disk I/O performance. From these results, block wiping technique with 32 KB buffer was adopted for subsequent iterations. The

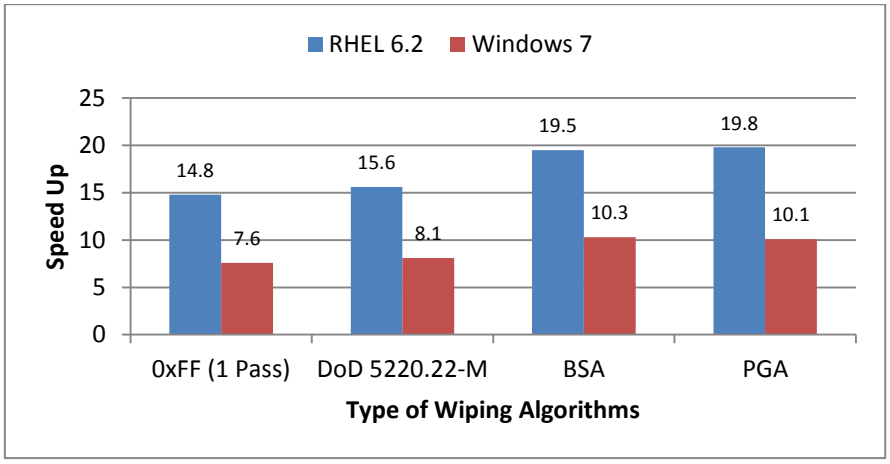
experimental result of applying *block buffer* optimization in the enhanced version of the data shredding software is shown in *Figure 3*.

Figure 3: Performance of disk I/O with various data block sizes



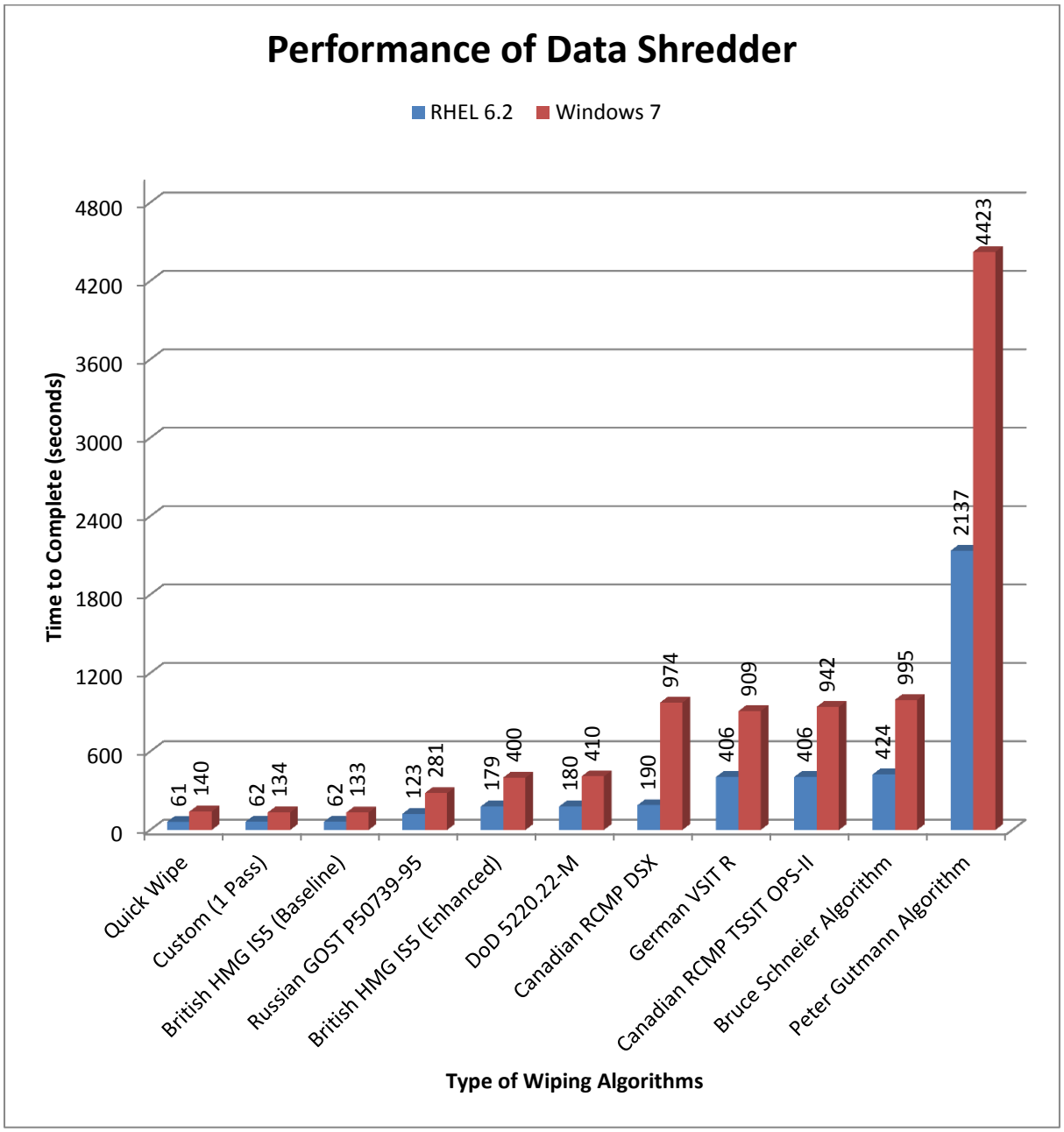
As indicated in *Figure 4*, adopting block buffer optimization improves the overall wipe performance of the DoD 5220.22-M method on an 80 GB hard disk drive by at least 8 times when compared with the original version of the CBL Data Shredder software written in C++ running exclusively on the Windows OS. The DoD 5220.22-M algorithm was used for this testing because it is one of the more popular standard used for data destruction (Speedie, 2009).

Figure 4: Performance of selected data destruction methods on an 80 GB hard disk drive using Windows 7 and Red Hat Enterprise Linux 6.2



Besides incrementing the size of the write buffer, the Python interpreters' *dynamic translation* (Just-In-Time compiler or JIT) method was also used to improve the runtime performance. The comparison of different data destruction methods on a 512 MB flash drive using Windows 7 and Red Hat Enterprise Linux (RHEL) 6.2 is shown in *Figure 5*.

Figure 5: Wipe performance of different data destruction methods on a 512 MB flash drive using Windows 7 and Red Hat Enterprise Linux 6.2



4.3 Comparison with other Open Source solutions

Table 3 compared our software with DBAN (Horn, 2012) and HDDErase. These are open source software commonly used for data destruction. However, DBAN supports only five commonly used data destruction algorithms. Moreover, the application has to be run at boot time. It denied users the opportunity to access their computer. Our software has a user-friendly graphical user interface (GUI) and offers a choice of ten different wiping algorithms inclusive of those that had been provided by DBAN. In lieu of multiple data storage devices present in a computer, we have engineered the software to sanitize multiple storage devices concurrently. The data

destruction process runs as a background task, thus permitting users to work on their computer.

Table 4: Comparison with other open source software

	DBAN	HDDErase	Our Software
Operating Systems	Windows, Linux	Windows	Windows, Mac OSX, Linux
Data Destruction Algorithm Support	• DoD 5220.22-M • RCMP TSSIT OPS-II • Gutmann's Algorithm • Random Data • Write Zero	• Secure Erase	• British HMG IS5-Baseline • British HMG IS5-Enhanced • DoD 5220.22-M • RCMP TSSIT OPS-II • RCMP DSX • German VSITR • Scheiner's Algorithm • Gutmann's Algorithm • Write Zero, One, Random • Write custom bit pattern
GUI	YES	NO	**YES**
Concurrently wipe multiple devices	NO	NO	**YES**
Ability to handle bad/corrupt hard disks	NO	YES	**YES**

5. Conclusion

We have developed data destruction software that ensures a secure and irrecoverable method to media sanitization. Several industry grade data destruction methodologies were incorporated in the application. In comparison with other open-source alternatives, our application was found to be more resilient. It could also support a wider choice of data destruction algorithms. Paid solutions may provide a similar functionality in terms of the algorithms. However, on comparing the speed of data destruction on an 80GB hard disk drive with the DoD 5220.22-M algorithm (Defense, 1995), our solution was found to be eight times faster. Further, no traces of deleted data were found upon verification with AccessData's Forensic Toolkit software and SHA-1 hash computations. Apart from ensuring cross-platform portability to reach maximum users, the software works well on solid state drives, non-volatile memory (NVRAM), USB flash drives, CD-RW/DVD-RW drives, as well as PATA/SATA/SCSI/SAS hard disk drives residing on laptops, desktop PCs, workstations, servers, and low-cost storage appliances. However, the application needs to be sufficiently tested on magnetic tape drives, Fibre Channel drives, large JBODs, NAS and SAN storage systems. Further investigations are desired if these categories of storage devices are to be addressed in the near future.

References

Bach, M., 1986. *The Design of the UNIX Operating System.* s.l.:Prentice Hall.

CGSecurity, 2012. *TestDisk.* [Online]
Available at: http://www.cgsecurity.org/wiki/TestDisk

Defense, U. D. o., 1995. *DoD 5220.22-M,* s.l.: National Industrial Security Program Operating Manual.

Deitel, H., 2004. *Operating Systems.* 3rd ed. s.l.:Pearson Prentice Hall.

Gutmann, P., 1996. *Secure Deletion of Data from Magnetic and Solid-State Memory.* s.l., USENIX Association.

Gutmann, P., 2001. *Data Remanence in Semiconductor Devices.* Berkeley, USENIX Association.

Horn, D., 2012. *DBAN, Hard Drive Disk Wipe and Data Clearing.* [Online]
Available at: http://dban.org/

Makoto , M. & Takuji, . N., 1998. Mersenne twister. *ACM Transactions on Modeling and Computer Simulation (TOMACS),* 8(1), pp. 3-30.

NIST, 2006. *Guidelines for Media Sanitization,* s.l.: U.S. National Institute of Standards and Technology.

Pirifrom, 2012. *Recuva.* [Online]
Available at: http://www.piriform.com/recuva

Precord, C., 2010. *wxpython 2.8 Application Developmentt Cookbook.* s.l.:Packt Publishing.

Reyes, A. & Wiles, J., 2007. *Cybercrime and Digital Forensics.* s.l.:Syngress Publishing.

Schneiderman, B., 2003. *Strategies for Effective Human-Computer Interaction.* s.l.:Pearson/Addison Wesley.

Speedie, A., 2009. *Choosing a secure Data Destruction Method,* s.l.: Secure I.T. Disposals Pvt. Ltd..

Technologies, C. D. R., 2012. *Data Recovery by CBL Hard drive and RAID recovery.* [Online]
Available at: http://www.cbldatarecovery.com/