A methodology for the repeatable forensic analysis of encrypted drives[*]

Cory Altheide[†]
IBM ISS
Mountain View, CA
caltheide@us.ibm.com

Claudio Merloni
Secure Network S.r.l.
Agrate Brianza, Italy
c.merloni@securenetwork.it

Stefano Zanero[‡]
DEI - Politecnico di Milano
Milano, Italy
zanero@elet.polimi.it

ABSTRACT

In this paper we propose a sound methodology to perform the forensic analysis of hard disks protected with whole-disk encryption software, supposing to be in possession of the appropriate encryption keys. We demonstrate how to create a forensically sound clone-copy of the seized media, and how to access the information contained in the media in a repeatable way, minimizing the usage of unverified and proprietary software. We discuss the impact of such encryption solutions on the capability of forensic analysis software to reconstruct deleted files. We propose and perform scientific tests for validating each step of our proposed procedure.

Categories and Subject Descriptors

K.5.m [**Legal Aspects of Computing**]: Miscellaneous—*computer forensics*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Unauthorized access (e.g., hacking, phreaking)*; E.5 [**Files**]: [Organization/structure]

General Terms

Documentation, Experimentation, Legal Aspects

Keywords

Computer forensics, whole disk encryption, cybercrime, data recovery.

1. INTRODUCTION

The concern for the security of data stored on lost or stolen laptops brings a growing number of organizations to the use of whole-disk encryption software. This, in turn, creates a number of issues for the forensic analysis of such encrypted media. A first issue is, obviously, that if the keys cannot be retrieved such evidence is not accessible to the analyst. Previous research has addressed this problem, suggesting to capture the keys from memory with live forensics [14], or through other strategies [7]. However, even once the keys have been obtained, a proper, standard procedure must be adopted to properly execute a forensically sound analysis of the seized encrypted media.

A common way to deal with this issue is to perform forensics in a live fashion [10], however in some legislative frameworks performing such an analysis may entail unwanted consequences, or may not be fully acceptable in a court of law because of the issue of repeatability. We found out that, on the other hand, a proper methodology for handling, decrypting and accessing the encrypted media once the key has been obtained was not documented or addressed in literature

Therefore, in this paper we propose a forensic methodology to perform such analysis, under the hypothesis that we are in possession of the keys. We discuss how to create a forensically sound clone-copy of the seized media and how to access the information stored inside such copy, while minimizing the usage of unverified, proprietary software. We discuss the impact of the use of encryption software and of our proposed procedure on the capability of forensic analysis software to reconstruct deleted files. We also propose tests that can be used for validation of each step of our proposed procedure.

The remainder of this document is structured as follows: in Section 2 we give a clear definition of the problem and the questions that we want to answer. In Section 3 we outline our proposed methodology. In Section 4 we illustrate our experimental validation results. Finally, in Section 5 we draw our conclusions.

2. PROBLEM STATEMENT

The problem we needed to address was to determine if and how it is possible to properly execute a forensically sound analysis of a seized computer, where the hard-disk has been protected with whole-disk encryption.

In particular, we were asked three questions (in the context of an investigation conducted by an Italian court):

1. Is it possible to create a forensically sound clone-copy of the seized media? Which procedure can be adopted to ensure integrity?

2. Is it possible to access the information contained in

[*]Proceedings of the ACM SIGOPS European Workshop on System Security (EUROSEC), Glasgow, Scotland, March 31, 2008

[†]Mr. Altheide was with Google Inc. during part of the work described in this paper.

[‡]Corresponding author.

the media? Is it possible to do so in a repeatable, scientifically sound way? Is it possible to ensure the integrity of the data extracted from such media?

3. What is the impact of such encryption software on forensic reconstruction techniques [5, 11], e.g. the retrieval and reconstruction of deleted files and data?

It is important to note that we had full access to decryption keys and passwords, as the party subject to seizure was cooperating with the law enforcement: this was not the core issue we were presented with.

In a case where the party subject to seizure is not willing (or is unable) to cooperate with law enforcement, before executing the procedures outlined below, keys would need to be retrieved, unless a vulnerable cryptographic algorithm, or a vulnerable implementation, had been used. Keys could be captured, for instance, in a live forensics fashion as suggested in [14, 3]. A recent paper also suggests that keys can be extracted from the hardware of seized computers, even if shut down, through an unconventional procedure [15]. In general, we found that a number of previous works have dealt with the issue of obtaining encryption keys, or at least noted it, while the issue of proper handling and access to encrypted media once the key has been obtained was completely forgotten.

It must also be noted that in the case of encryption applied to files (as opposed to whole media) detecting that a file is encrypted, and not simple gibberish, obtaining the keys, and obtaining an appropriate decryption tool is everything that a forensic analyst needs to do, and this case has already been examined by a number of authors [21, 7, 13] which all offer more or less the same suggestions. In some cases these procedures have been intuitively extended to encrypted partitions or drives [9, 8], but without considering the additional questions on the integrity of the evidence that this shift entails. For this reason we set out to create a sound and documented methodology for obtaining a plaintext version of the disk, which was evidently missing in literature.

3. PROPOSED METHODOLOGY

3.1 Considerations on cloning

Creating a forensically sound clone copy of the evidence, in order to minimize the handling of the original, is an extremely well established procedure [22, 17, 20, 11]. Cloning an encrypted media is not different, provided that the encryption has not been applied at hardware level. After properly connecting the original media to a computer, using write-blocking hardware, using an open source operating system (completely documented and analyzable in any part), and open source software validated for forensic use, the proper sequence of operations is:

- hashing of the source device
- clone-copy of the source device to a bitstream image
- hashing of both the source device and the bitstream image

The checksums at the end of the process should be all correspondent and correspondent to the original checksums of the source devices, proving that the evidence has not been altered, and that the clone-copy is identical to the original evidence. From this point on, every operation can and should be conducted on the clone copy.

If a hardware encryption device has been used the problem of how to acquire a bitstream image for forensic use is more complex. For instance, this could very well be the case with applications of Trusted Computing in the near future [3]. In [16] the authors, who are discussing the implications of such technology on forensic investigations, fail to recognize it as a problem. A solution of this problem is beside the scope of this paper, but it will be the objective of our future work in this area.

3.2 Extraction of data: open source software vs. proprietary software

As it is well known from literature, it is important in forensic analysis to use only software and hardware which has been properly and thoroughly validated for forensic use [12]. In particular, the use of open source software [19, 4, 18] is commonly recommended in the academic world, since it makes every step of the reconstruction process verifiable. On the other hand, proprietary software can lead to verifiable results, provided that the manufacturer gives a thorough documentation and test cases demonstrating what their software does, and that the forensic community is made able to verify their claims (e.g. through disclosure of the source code of the proprietary tools).

However, if a proprietary software solution for whole-disk encryption is used, the forensic analyst has no option but to use the corresponding closed source, proprietary application suites to get a cleartext bit-stream image of the media for further analysis, if possible; or in the worst case, the analyst needs to explore the image from within, using the decryption software itself to access the data. In neither case the vendor is likely to provide documentation of forensic soundness of their proprietary tools. Without exploring this issue (which would lead us away from the focus of the paper), we may note that customers who are deploying a full-disc encryption solution should ask their vendor an appropriate answer to these issues, with certified procedures and verifiable software, in order to avoid future problems.

Even in the case that encryption algorithms and formats used are public (as it is the case, e.g., for SafeGuard Easy), decrypting such a disk without resorting to the vendor's own software is typically unfeasible, as it would entail an intense reverse engineering work to create a suitable decryptor based on the available information.

Of course, the principles of minimality and necessity of the interaction with the evidence should apply here: therefore, we need to establish a procedure which minimizes the use of proprietary and unverified software. We also need to make the access and decryption repeatable, and to verify insofar as possible the correspondence between the encrypted original and the decrypted version of the file system.

In many cases (as it is the case with the software we analyzed), the vendor offers a way to permanently decrypt a drive. Usually, this comes in the form of a boot disk which can be used, along with the decryption keys of the user or with a master decryption key, to execute the decryption. If this is the case, it is possible to use the following procedure to obtain a cleartext version of the media:

- clone-copy of the encrypted bitstream image to a work image;

- hashing of both images for verification;
- using appropriate tools, the work image can be converted to a format suitable for execution in a virtual machine environment; using as an example the open source virtual machine *qemu*, it is possible to use directly the raw images; if *vmware* is used instead, it is possible to use the open source tool *qemu-img* to convert the raw image to the vmdk format;
- configure the chosen virtual machine to boot a copy of the emergency disk of the encryption technology in use, using the transformed work copy of the image as a connected virtual drive; it is important to check the boot order of the virtual machine to ensure that it correctly boots the emergency disk and not the work image;
- boot the virtualized system, and decrypt the virtualized "hard disk"; the procedure obviously depends on the encryption technology in use;
- at this point, the decrypted bitstream image should be converted back to the raw format and hashed, for future reference.

This sequence of operations is repeatable at any time, with a deterministic result, limiting the usage of a non-verified proprietary software to the strict minimum.

At this point we may wish to prove that the decrypted version of the image has the same contents of the original system, if it were accessed normally. This implies similar challenges [6] to ensuring correctness in a live forensics setup [2]. Since the "normal" access to the original system entails booting it, it is not possible to ensure a bitwise correspondence. This is an intrinsic limit imposed by the presence of whole-disk encryption: there is no better way to create a "plaintext copy" of the encrypted image, because actually there is no "plaintext original": encryption and decryption happens on-the-fly in this type of software, so a plaintext version of the image did never actually exist. So, the fact that we cannot ensure "correspondence" of the plaintext image to an ideal "plaintext original" is not really meaningful. What we can do is to ensure a file-by-file correspondence between the decrypted image and the image running inside the live, booted system, excluding the files that are modified during the boot process of the system itself. This can be demonstrated through the use of a recursive MD5 calculation.

Similar issues have been explored in [10] while evaluating the usefulness of recreating an acquired image in virtual machine environment. A virtual machine approach allows booting the acquired image to an almost identical environment as the originally investigated machine thus allowing for a fast and efficient analysis. The authors propose a verification of the findings using more traditional methods; however, in our case such a verification is impossible, as there is no "forensically preserved original" against which to compare.

In the unlikely case that the vendor does not offer any way to permanently decrypt a drive (i.e., the encryption of the drive is a non reversible procedure), no procedure can be used to create a plaintext bitstream image of the media. In this case, the only possible way to access the media is to perform a live forensics as described in [2, 6] on the work image created as described above, and booted inside a virtual machine environment. The sequence of operations will be repeatable, but the correctness of the operation will suffer of the usual degradation implied by live forensics. However, this is once more an intrinsic limit which cannot be bypassed.

In the case of non-system disk encryption, it is possible that the decryption software comes in the form of a standalone software, as opposed to a boot disk. The procedure in this case would be similar:

- clone-copy of the encrypted bitstream image to a work image;
- hashing of both images for verification;
- appropriate connection of the work image on the host operating system
- execution of the tool to decrypt the work image; also in this case, the procedure depends on the encryption technology in use;
- dump of a bitstream image of the decrypted work image and hashing, for future reference.

Once again, it may be impossible to prove that the decrypted version of the image has the same contents of the original drive, if it were accessed normally while mounted on the original host operating system. Since the "normal" access would be mediated by the decryption software, it is probably impossible to ensure a bitwise correspondence; once again a recursive MD5 calculation can be used.

This sequence of operation also incidentally demonstrates that as far as the encryption is purely software-based, the original hardware is not needed for any of the operations. Actually, unless the legislation we are operating under otherwise requires to keep the original hardware under seizure, once the clone-copy of the encrypted media has been obtained, seized hardware can be safely returned to the owner. This is something we were asked to demonstrate under Italian procedures for evidence.

3.3 Impact on data reconstruction capabilities

In order to understand the impact that this decryption phase has on the classic forensic tasks of deleted files reconstruction and recovery, we suggest to take a statistical approach. We can create, on a test machine, an encrypted image using the particular software in use. Then we can create, and delete, a number of test files of known content. After executing the whole procedure, we can then analyze the decrypted image with any forensic software of choice and verify how many of these file are retrievable. Ideally, almost all of them should be retrievable, except the ones overwritten by the shutdown procedure of the test machine. If that is the case, we can conclude that the process does not, per se, negatively interact with forensic recovery and reconstruction of deleted files, which is already a statistical game of luck. This is intuitively true, however, since any whole-disk encryption software, as they must be as agnostic as possible with respect to the operating system working on top of them.

4. EXPERIMENTAL VALIDATION

As a side note, the analyzed machine was a Lenovo ThinkPad T43P laptop, and the disk encryption software was a proprietary tool, specifically SafeGuard Easy (version 4.40.2),

Table 1: Matching and non matching hashes in the decrypted file system vs. the live filesystem

	Number	Percentage
Matching	44423	96.66 %
Non-matching	1535	3.34 %
of which automatic backups	1445	3.14 %
of which other system files	90	0.20 %
Total	**45958**	**100%**

produced by Utimaco Software AG. The specific hardware and software is not relevant, however, while it is relevant to distinguish between the procedure in case of an open software, and in the case of a proprietary software, as we will see below. Hardware based cryptography could introduce other issues.

We used, for our experiments, two GNU/Linux "live" distributions, specifically aimed to forensics, BackTrack v2 and Helix v1.9a. On such platforms, hashing can be obtained by use of the standard `md5sum` and `sha1sum` command-line utilities, while bitwise copy can be obtained by using the `dd` tool, which has been validated for this use by the NIST [1].

We installed Windows XP on the laptop, and installed the encryption software, following the documented procedures. The disk was partitioned and formatted with a single partition using the NTFS file system. We activated the encryption functionality on said partition and rebooted the machine. We proceeded to create a small number of text files with known content, for test purposes. In particular, we created 60 test files containing the texts of Homer's "Iliad" and "Odissey", and of Dante's "Commedia" (20 each). Using the `md5deep.exe` tool (`http://md5deep.sourceforge.net/`) we computed the MD5 hash of each file present, on the live (booted) system. Of course, a handful of system files could not be accessed as they are locked by the OS when the system is booted. We then deleted the test files, and emptied the "Recycle Bin".

We successfully applied the procedure in Section 3.1 to obtain a clone copy of the drive, and then we proceeded to create a plaintext image as described in 3.2, using both `qemu` and `vmware` with identical results.

We computed the MD5 of each file in the plaintext image, using the `md5sum` tool contained in Helix, accessing the image in read-only mode. We compared the results with the list of MD5 obtained from the live system, and we report in Table 1 some statistics on the number of files with a matching hash. The non-matching files are either temporary or system files, as expected: in particular, 1445 out of 1535 files are from the directory `c:\system volume information\_restore*`, which is an automatic backup of system files that Windows creates at boot. The small number of files, and the fact that they are all involved in the boot process, allows to conclude that the process of conversion to plaintext does not create any significant alteration.

We then proceeded to analyze the image with the Sleuthkit toolkit, in particular using the `autopsy` forensic browser. We were able to retrieve almost all the test files created and deleted on the original device. We can therefore conclude that the process does not, per se, negatively interact with forensic recovery and reconstruction of deleted files. We can also conclude that in the specific case of SafeGuard Easy these capabilities are preserved, as intuition predicted.

5. CONCLUSIONS

We addressed the problem of creating a methodology to perform a forensically sound analysis of a seized computer, where the hard disk has been protected with whole-disk encryption. We supposed to be in possession of proper decryption keys for the media, and focused on the procedures for properly executing the analysis. In particular, we demonstrated how it is possible to create a forensically sound clone-copy of the seized media, ensuring evidence integrity, in any framework which is purely software based. We demonstrated a way to access the information contained in the media in a repeatable, scientifically sound way, minimizing the usage of unverified, proprietary software. We also incidentally demonstrated that, as far as the encryption is purely software-based, the original hardware is not needed for any of the operations besides the first extraction of the clone-copy of the encrypted media. We discussed the impact of such encryption software on forensic reconstruction techniques. We proposed scientific tests for validating each step of our proposed procedure, and we also experimentally demonstrated that our approach work on a specific test setup.

Future extensions of this work may try to address issues such as appropriate forensics analysis procedures in the case of hardware-based encryption. We are also trying to perform more extensive tests with other types of encryption software and host filesystems and operating systems, to better demonstrate the general validity of our methodology, and to discover if in some cases the procedure impacts the reconstruction capabilities of forensic software.

Acknowledgments

The authors gratefully acknowledge the support of Mr. John Flynn of Google Inc., and Mr. Manuel Pellizzon, of Google Italy S.r.l.; we also wish to acknowledge the helpful comments of Mr. Giuseppe Vaciago, a lawyer based in Milan, Italy.

6. REFERENCES

[1] Test results for disk imaging tools: dd gnu fileutils 4.0.36, provided with red hat linux 7.1. Technical report, National Institue of Justice, August 2002.

[2] F. Adelstein. Live forensics: diagnosing your system without killing it first. *Commun. ACM*, 49(2):63–66, 2006.

[3] M. Burmester and J. Mulholland. The advent of trusted computing: implications for digital forensics. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 283–287, New York, NY, USA, 2006. ACM.

[4] B. Carrier. Open source digital forensic tools: The legal argument. Technical report, @stake Research Report, October 2002.

[5] B. Carrier. *File System Forensic Analysis.* Addison-Wesley Professional, 2005.

[6] B. D. Carrier. Risks of live digital forensic analysis. *Commun. ACM*, 49(2):56–61, 2006.

[7] E. Casey. Practical Approaches to Recovering Encrypted Digital Evidence. *International Journal of Digital Evidence*, 1(3), 2002.

[8] J. Craiger, M. Pollitt, and J. Swauger. *Law*

Enforcement and Digital Evidence. John Wiley & Sons, 2005.

[9] J. Craiger, J. Swauger, and C. Marberry. Digital evidence obfuscation: recovery techniques. In *Proceedings of SPIE, the International Society for Optical Engineering*. SPIE.

[10] E. H. Derek Bem. Computer forensic analysis in a virtual environment. *International Journal of Digital Evidence*, 6(2), 2007.

[11] D. Farmer and W. Venema. *Forensic Discovery*. Addison Wesley Professional, 2004.

[12] G. E. Fisher. Computer forensics tools verification. Technical report, NIST. available online at http://www.itl.nist.gov/div897/docs/computer_forensics_tools_verification.html.

[13] S. Garfinkel. Anti-Forensics: Techniques, Detection and Countermeasures. In *Proceedings of the 2nd International Conference on i-Warfare and Security (ICIW)*, pages 8–9, 2007.

[14] I. Golden G. Richard and V. Roussev. Next-generation digital forensics. *Commun. ACM*, 49(2):76–80, 2006.

[15] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: Cold boot attacks on encryption keys. submitted for publication, February 2008.

[16] M. Hannan and T. Wilsdon. The implications of hardware encryption devices on forensic computing investigations. In A. Jones, editor, *Proceedings of the 3rd European Conference on Information Warfare and Security*, 2004.

[17] K. J. Jones, R. Bejtlich, and C. W. Rose. *Real Digital Forensics: Computer Security and Incident Response*. Addison-Wesley Professional, 2005.

[18] E. E. Kenneally. Gatekeeping out of the box: Open source software as a mechanism to assess reliability for digital evidence. *Virginia Journal of Law and Technology*, 6(1), 2001.

[19] D. Manson, A. Carlin, S. Ramos, A. Gyger, M. Kaufman, and J. Treichelt. Is the open way a better way? digital forensics using open source tools. In *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, page 266b, Washington, DC, USA, 2007. IEEE Computer Society.

[20] G. M. Mohay, A. Anderson, B. Collie, R. D. McKemmish, and O. de Vel. *Computer and Intrusion Forensics*. Artech House, Inc., Norwood, MA, USA, 2003.

[21] J. Siegfried, C. Siedsma, B.-J. Countryman, and C. D. Hosmer. Examining the encryption threat. *International Journal of Digital Evidence*, 2(3), 2004.

[22] J. R. Vacca. *Computer Forensics: Computer Crime Scene Investigation (Networking Series) (Networking Series)*. Charles River Media, Inc., Rockland, MA, USA, 2005.