WORM is not enough!

Soumyadeb Mitra

Department of Computer Science, UIUC

Abstract

Important documents like financial reports, customer communications etc are increasingly being maintained by businesses in electronic format. These represent much of the data on which key decisions in business operations are based and hence must be maintained in a trustworthy fashion - safe from destruction or clandestine modification. Secure retention of such data is also increasingly being regulated by govt regulations like Sarbanes-Oxley Act or SEC Rule 17a 4. Thus there has been a recent rush to introduce Write-Once-Read-Many (WORM) storage devices. In this paper, we argue that simply storing records in WORM storage, as is the current focus, is far from adequate to ensure that the records are trustworthy. The key issue is that for data to be truly trustworthy its entire lifecycle has to be secured: starting from the process of creating it, to storing & maintaining it and finally retrieving. In this paper we show that it is possible to compromise both the maintenance and retrieval of records even if it is maintained on WORM.

1. Introduction

Records such as electronic mail, financial statements, medical images, drug development logs, quality assurance documents and purchase orders are valuable assets. They represent much of the data on which key decisions in business operations and other critical activities are based. Having records that are accurate and readily accessible is therefore vital. Records also serve as evidence of activity, but to be effective the records must be credible and accessible. Given the high stakes involved, tampering with the records could yield huge losses and must be specifically guarded against, especially as the records are increasingly in electronic form, which makes them relatively easy to delete and modify without leaving so much as a trace. Ensuring that the records are trustworthy, i.e., not only readily accessible and accurate, but also credible and irrefutable, is particularly imperative in the litigious US. On average, a Fortune 500 company is the target of 125 non-frivolous lawsuits at any given time, and the damages awarded are increasingly rapidly, as is the cost of electronic data discovery, which is projected to rise at 65% per year to reach $2 billion in 2006 [9].

Furthermore, a growing proportion of the records are subject to regulations that specify how they should be managed. In the US alone, there are currently more than 10,000 such regulations [11]. The key focus of many of these regulations (*e.g.,* Sarbanes-Oxley Act [2], SEC Rule 17a-4 [8]) is to ensure that records are trustworthy. Non-compliance with these regulations could result in stiff

penalties. For example, unprecedented fines have recently been levied by several regulatory bodies, including the Securities Exchange Commission (SEC) and the Food and Drug Administration (FDA), for non-compliance. The bad publicity of non-compliance and the ensuing investor flight alone could cost an organization dearly. As information becomes more valuable to organizations and with recent headlines of corporate misdeeds, accounting scandals and securities fraud, the number and scope of such regulations are likely to grow. Worldwide, the volume of compliant records is projected to increase by 64% per year to almost 2 PB in 2006 [9].

The key requirement for ensuring trustworthiness of data is to ensure that it is permanent - data must be saved to media that cannot be altered or erased until a specified expiration date. The data permanence requirement is particularly important in the financial industry as a result of heightened scrutiny by the SEC and other law enforcement authorities. SEC Rule 17a-4 (see Appendix) mandates data permanence for "all communication (internal or external) related to the business as such." Thus, there has been a recent rush to introduce Write-Once-Read-Many (WORM) storage devices to enable the effective preservation of records. Storage vendors [3,6,7,10] have come up with different WORM solutions and these are being widely embraced by the industry.

In this paper we content that simply storing the records on a WORM device, as is the current focus, is far from adequate to ensure that the data is trustworthy. For data to be truly trustworthy its entire lifecycle has to be secured: starting from the process of creating it, to storing & maintaining it and finally retrieving it.

The key issue is that with today's large volume of records and increasingly stringent query response time [9], some form of direct access mechanism such as an index must be available as needed to access the records. Unless this index is also maintained in a proper fashion, a record stored in WORM storage can in effect be hidden or altered, by simply modifying the index entry pointing to that record, as illustrated in figure 1.

The index too hence must be maintained on Write-Once media. In this paper we argue that the almost all the current generation WORM devices are not suitable for maintaining indexes. Hence businesses usually take the approach of creating the index on a separate read-write media, while storing the actual data on WORM [12]. However as explained above this is not secure.

The second big challenge is storing and maintaining records trustworthily, over very periods of time (often a couple of decades) as is required by some of the compliance regulations. A WORM device ensures that a record once committed to it cannot be tampered with. It is, however, not clear if a record can be maintained on a single device for such extended periods of time. Practical considerations like failure of devices, obsolete technology etc might require the records to be migrated from one WORM device to another over its lifetime. This however introduces a weak link in the process of ensuring trustworthiness, by giving the adversary an opportunity to delete or modify records during migration. It is particularly a concern if the adversary is a company insider,

like the CEO or CFO (more often than not, the biggest threat to data comes from company insiders, including people at the highest level) who has enough privilege to initiate and control the migration process and to modify or omit records during migration. In this paper, we argue that migration is likely to be common operation in typical business and techniques must be developed for ensuring trustworthy migration.

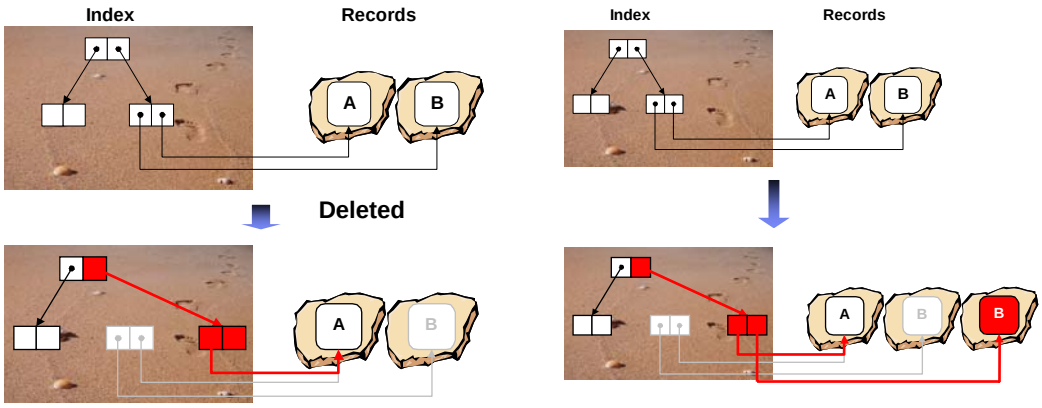


Figure 1 shows that record B cam be effectively hidden or deleted by modifying the index entry pointing to its

.

This paper is organized as follows. We start with giving an overview of the two popular WORM devices in section 2, EMCs Centera [3] and Network Appliances SnapLock [7]. Although both implement Write Once semantics, their external APIs are very different: While EMC offers an object interface which allows applications to commit objects, network appliance provides a conventional file based interface. We discuss these APIs and the underlying technology in details.
In section 3, we give an overview of a popular indexing data structures : Inverted Index. Inverted Indexes are used for keyword based queries where a user gives a list of keywords and gets documents containing those keywords in the result. These indexes are used for searching unstructured data like emails, notes etc. In section 4 we explain why the current generation WORM devices are not suitable for maintaining Inverted Index.
In section 5, we discuss the problem of secure migration. We motivate the problem by discussing a couple of realistic scenarios which might require data-migration. We present a threat model and briefly discuss why some of the common techniques (including those supported by the storage vendors are not adequate).

2. Write Once Devices

2.1. Network Appliance's SnapLock Technology

Network Appliances WORM solution is implemented as a software extension to their Network Attached Storage or NAS devices. The term NAS refers to a storage device that is connected to a network (usually TCP/IP) and provides remote file access service. The end hosts access the files stored on a NAS storage using common file access protocols such as NFS or CISF. To an end host, a NAS device hence appears as a NFS or a Windows file server.

Internally, a NAS storage device consists of an engine which implements the remote file services (NFS/CISF server) and manages all the devices, on which data is stored. The storage engine used inside Network Appliance's NAS devices is their proprietary operating system, called the Data ONTAP microkernel.

The data ONTAP kernel has a special feature which lets users mark certain portions of the storage read-only. Users are only allowed to create and add to files in these volumes - Any modification or deletion operation on files created in this volume is disallowed. This effectively provides the WORM functionality.

2.2. EMC Centera

EMC Centera is based on a novel storage architecture known as content addressable storage or CAS. In a CAS storage, the actual contents of a file is used to derive its location. Traditionally storage devices (like the hard-disk connected to a PC or remote network attached storage in a lab) are location addressable. In a location-addressable storage device, each element of data (for example a file) is stored in some location on disk and this location is recorded for later use. The location of a file is recorded inside its parent directory, along with some additional meta-data like file name, owner, permissions, size and so on. When a future request is made for a particular file, this directory is searched to obtain its location on disk, from where it is retrieved. When a new file is created on a location-addressed device, it is simply stored in the next available free location on the storage and its location is recorded in the parent directory. When a file is modified, it is fetched from disk, updated and stored back in the same location. If some new data is added to an existing file, the data is either stored at the end of its current location (if space is available) or in the next free location on disk: In the later case, the address of the new location is also recorded in the directory. A file hence can span multiple locations on disk.

In contrast, when a file is stored into a CAS system like EMC Centera, instead of storing the file in the next available free space, its content is used to derive its location: One still needs to maintain a mapping between filename and its location on disk: However the location now is implicit: It is solely a function of the contents of the file.

CAS storage is suitable for fixed content documents like emails, financial records, audit reports, X-Ray images etc. CAS provides an assurance that the retrieved document is identical to the one originally store - If the documents were different, their content addresses would differ.

The storage architecture of EMC Centera consists of a series of networked nodes, divided between **storage** nodes and **access** nodes. The access nodes maintain a synchronized directory of content addresses, and the corresponding storage node where each address can be found. When a new data element, or **blob**, is added, the device calculates a hash of the content and returns this hash as the blob's content address. As mentioned above, the hash is looked to verify that identical content is not already present. If the content already exists, the device does not need to perform any additional steps; the content address already points to the proper content. Otherwise, the data is passed off to a storage node and written to the physical media.

When a content address is provided to the device, it first queries the directory for the physical location of the specified content address. The information is then retrieved from a storage node, and the actual hash of the data recomputed and verified. Once this is complete, the device can supply the requested data to the client. Within the Centera system, each content address actually represents a number of distinct data blobs, as well as optional metadata. **Whenever a client adds an additional blob to an existing content block, the system must recompute the content address and move the existing data to the new address.** Unfortunately this can be extremely inefficient: For example even if a byte is appended to a 1 gigabyte file, the entire file must be moved. CAS devices hence employ various optimization techniques like storage virtualization [1] to prevent this data copying step. Unfortunately, even with these optimizations, CAS devices are not very as efficient as conventional NAS based devices in supporting file append operation (for example, the hash still needs to be computed). In the next section we discuss why this is a problem for maintaining dynamic index structures like Inverted Indexes on WORM.

3. Indexing Structures

3.1. Inverted Index

Inverted Indexes are used to support keyword based queries. An inverted index consists of a dictionary of keywords and a list of document identifiers stored with each keyword: The document identifiers refer to documents which have that particular keyword. These lists are called the

posting list. Figure 2 illustrates an example inverted index. The figure shows that the keyword "Hello" is present in documents with identifiers 41, 55 and 89 and so on for the other words.

Keyword queries are answered by scanning the posting lists of the keywords, thereby obtaining a list of documents that have the keyword. Multi-keyword queries (queries in which all the keywords must be present) are answered by taking an intersection of the posting lists. For example a query "**Hello World**" (referring to all documents which have both words **Hello** and **World**) is answered by taking the intersection of **Hello** and **World**'s posting list: Only document identifier 89 lies in the intersection.

When a new document is added to the system, it is first assigned a next available document identifier - Document identifiers are usually assigned through an increasing counter. The document is then parsed to extract the list of keywords contained in it and its document identifier is appended to the posting list of all those keywords. Adding a new document hence requires the ability to append new entries to individual posting lists.

The individual posting lists can be maintained can be implemented on WORM device by maintaining each list as a separate object or file which can be appended to. Unfortunately this operation can be prohibitively slow on the current WORM devices, as we discuss below by considering both the EMC and Network Appliances WORM devices.

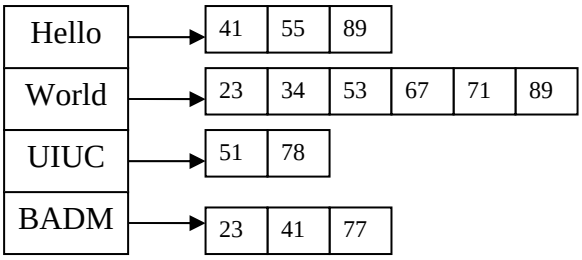


Fig 2: Example of an Inverted Index

3.2 Inverted Index on EMC Centera

EMC Centera is a content addressable device: When some data is appended to an existing file (or blob) the system has to recompute the content address. This operation can be prohibitively slow, particularly if the appends are frequent, as would be the case with updating the inverted index.

As an example, consider a typical email document: Emails without attachments have about 10-20 keywords (sender/ receiver(s) address, time, subject words etc) which can be queried upon and

must be indexed. Hence for every new email its document identifier must be appended to 10-20 different posting list.

Each such update would require re-computing a new address for the corresponding posting list. If this requires hashing the entire file content (no details was available from EMC about their algorithm used), this would be extremely slow. One can possibly optimize by computing the hash incrementally: The new hash can be computed by hashing together the old hash and the new data. Even with this optimization, the achieved update speed would typically be inadequate to support typical business document workloads. For example a single SHA1 (a popular hash function) computation on a number takes 100 milliseconds on commodity PC. Considering that each email would require 10-20 hash computations, this gives 1-2 sec of processing time per-email. In other words, the index would be able to handle only ½ to 1 emails per second, a rate inadequate even for small businesses.

Apart from re-computing the hash, the document identifier must also be appended to the actual disk. This operation can also be prohibitively slow, as described in the next section.

3.3. Inverted Index on Network Appliances

As explained earlier, Network Appliances storage systems export a filesystem interface to the external world allowing users to append to existing files. Appending to existing file does not suffer from the address-recomputation overhead as EMC Centera's. However updating the index in place can still be prohibitively slow because of the underlying disk technology. We first give an overview of current disk technology and discuss why updating the index can be a problem.

Hard-disk consist of a set of circular platters on which data is magnetically stored. There is a read-write head for each platter. In order to read or write data in a particular place on the disk, the read/write head of the disk needs to be moved to the correct place. This process is known as "seeking", and the time it takes to move head is the "seek time". Seek time dependent on disk mechanics like the size of the platters, the speed with which the read-write head can be moved, the accuracy with which it can be positioned etc. Unfortunately mechanical improvements of the disk structure have almost hit the wall: For example, although disk storage densities have improved impressively (60% to 130% compounded annually), seek time improvements have been occurring at only about 7% to 10% over the last decade. Current generation disk drives have average seek times of 8 milliseconds and this often is the dominating factor in the time it takes to fetch data from disk[1].

[1] There is another component called the latency. Latency is the time it takes to rotate the platter to position the head on the correct location. For our discussion we can assume that the latency is also a part of the "seeking" process.

Seeking is required only when the data to be read or written is not next to the current location of the read/write head. File-systems and databases hence try to organize data to reduce the expected seek time. One common strategy is to organize related data (data which is likely to be accessed together) close to each other on disk to reduce the effective seek time. This however is very data-structure/workload specific and not always possible.

Inverted Index is a good example of a data-structure which is not very suitable for such seek time optimizations. When posting lists are organized as separate files, an append operation to each posting list would require moving to the end of the file from the current position and hence would incur a disk seek. Considering the email example, if each email has 20 keywords, updating the index would require 20 different posting lists to be updated, each requiring a disk seek. Assuming a seek time of 8 milliseconds, the seeking overhead would be 160 millisecond per email. In other words, the index would be able to handle about 6 emails per second, which again is inadequate for typical businesses. In this example we only considered emails having 20 keywords. If the email attachments are also indexed, the number of keywords per email can be much higher.

In the last two sections we showed that both object storage or NAS storage are not very suitable for maintaining inverted indexes.

3.4. Current Approaches and Possible Solution

The last section showed that it is not feasible to efficiently update inverted indexes on disk. The argument given (high disk seek time) however is applicable to any disk based device, not just Write-Once media.

This problem has received a lot of interest in the research community and industry, because of the popularity of inverted indexes[2] and a number of solutions have been proposed for efficiently maintaining inverted indexes. This section gives an overview of two popular approaches and discusses why these techniques too are not feasible if the index is maintained on current WORM devices. One popular approach usually taken “re-build” the index from scratch. Instead of updating the posting lists in place (as we were doing earlier) the keywords of the new documents are appended to a log. Appending to log does not incur a disk-seek since data is always added to the same position (tail of the log) on disk. A new index is created from scratch by sorting this log. Sorting can usually be done very efficiently. Unfortunately this approach is not feasible for WORM: Creating a new index every time is space inefficient since the old index space cannot be reclaimed (WORM does not allow deletions).

[2] Search engines use inverted indexes

An alternate strategy often used is to buffer index updates in memory and commit the updates in batch to the index. Buffering helps in amortizing the disk-seek cost over multiple updates: Instead of paying one disk seek per posting list update, one can wait and collect enough updates and append them in one go. The disk seek time hence gets shared amongst multiple updates, thereby reducing the effective seek time per update. As an example, suppose we have to commit 100 emails, all having the same set of 20 keywords. If we commit each email separately, we would need 20*100 random disk seeks. On the other hand, if we buffer the 100 emails and update the posting lists of the keywords with all the 100 documents together, we need to pay only 20 disk seek operations (one for each posting list). This is a 100 times improvement over the trivial case.

Unfortunately it is unreasonable to assume that all the 100 emails will have the same set of 20 keywords. In a full text inverted index, where dictionary words are also indexed, the number of keywords is usually 100,000 or more. The amount of buffering required to effectively reduce the disk seek overhead is hence much higher in that case. For example in [13] authors had to buffer 100,000 documents to have a mere document update rate of 10 documents per second.

Buffering raises a serious security issue: Is it safe to leave such a huge gap between when the document is committed to WORM and when the actual index is updated? A malicious attacker (particularly if he is a company insider) can get rid of the index entry while it is still in the buffer, for example by crashing the application and deleting recovery logs[3].

In practice, any delay between when the a document is created and when it is committed on WORM can introduce unnecessary risk and complexity into the compliance process. For example, the prevailing interpretation of email retention regulations is that a regulated email must be committed as a record before it is delivered to a mailbox. If the email is delivered to a mailbox and read before it has been committed as a record, then the organization would be risking non-compliance and would become seriously exposed if it turned out that the record did not get committed.

There are two ways to address this problem. One option is to build the indexing capability inside the WORM devices. As explained earlier, the current WORM devices are built on top of normal read-write magnetic disk drives. The write-once semantics is enforced by the operating software running inside the device: It ensures that any external application is not allowed to delete or alter files. The operating software however has read-write access to the underlying device and hence can use either of the above two strategies (logging or buffering) to maintain the index efficiently. Buffering if done inside the WORM is safe: The operating software can ensure that external applications do not have access to the buffer and hence cannot remove entries from it. The problem with this approach is that this makes the storage device very complex. Furthermore, this

[3] Keeping the recovery logs or buffer on WORM is also not a solution - Eve can append commit or end of log markers to fool the application into believing that no recovery is required

scheme is not very adaptable. For example it is usually much easier to upgrade an indexing application (say to add support for new file formats) which is running outside the storage, as compared to an application which comes hard-wired inside the storage server.
We are currently developing techniques for building inverted indexes efficiently.

4. Another weak link: Data Migration

Just storing data on WORM is not adequate to ensure trustworthiness of data. For data to be truly trustworthy its entire lifecycle has to be secured: starting from the process of creating it, to storing & maintaining it and finally retrieving. A big challenge in this respect is storing and maintaining records trustworthily, over very large periods of time (often a couple of decades) as is required by some of the compliance regulations. A WORM device ensures that a record once committed to it cannot be tampered with. It is, however, not clear if a record can be maintained on a single device for such long periods of time. Practical considerations like failure of devices, obsolete technology etc might require the records to be migrated from one WORM device to another over its lifetime. This however introduces a weak link in the process of ensuring trustworthiness, by giving the adversary an opportunity to delete or modify records during migration. It is particularly a concern if the adversary is a company insider, like the CEO or CFO (more often than not, the biggest threat to data comes from company insiders, including people at the highest level) who has enough privilege to initiate and control the migration process and to modify or omit records during migration. In the next section, we discuss a couple of realistic scenarios which would force data migration. We then discuss why conventional migrations strategies are not adequate for ensuring trustworthiness.

4.1 Data Migration Scenarios

4.1.1 Obsolete Technologies

The timelines over which businesses are required to maintain records is often much larger than average lifespan of a storage devices or even IT technology. For example HIPAA requires medical records of infants to be maintained till they are attain the age of 21. Similarly OSHA requires employers to keep records of both medical and other employees who are exposed to toxic substances and harmful agents for 30 years. Devices like hard-disks, storage controllers etc , on the other hand ,do not have life spans extending over a half a decade or so. Also from past experience, one can speculate that even the underlying technologies, be it access protocols like iSCSI/Fibre Channel or the storage technology like IDE, SCSI might become obsolete in a decade or so. Accessing data from outdates storage devices would require companies to

maintain obsolete hardware and software and is likely to be costly (maintaining unsupported hardware or software would require specially skilled staff), error-prone (as it would be difficult to find replacement hardware, for example if RAID disk rashes, it might not be possible to get a new one to rebuild the RAID) and insecure (unsupported hardware and software might have bugs). The only way to address this problem is to migrate records from old WORM devices to new ones, as the technology evolves. Such migrations are also accompanied by data consolidation: Data from multiple old WORM devices are merged together into a single device.

Another related scenario that might require migration is while recovering from device failures. Businesses usually maintain two or more copies of data: A common usage scenario is to maintain a primary copy on fast but costly magnetic WORM device, and a secondary copy on slow by cheap optical WORM device. The primary copy is used for online use, while the secondary for special cases like recovering failures. For example, if the primary fails, data is recovered from the secondary optical device. Since user queries are answered only from the primary copy, without considering the secondary (accessing the secondary is slow and costly for regular querying), the migration from secondary from primary must be secured.

4.1.2 Security Considerations

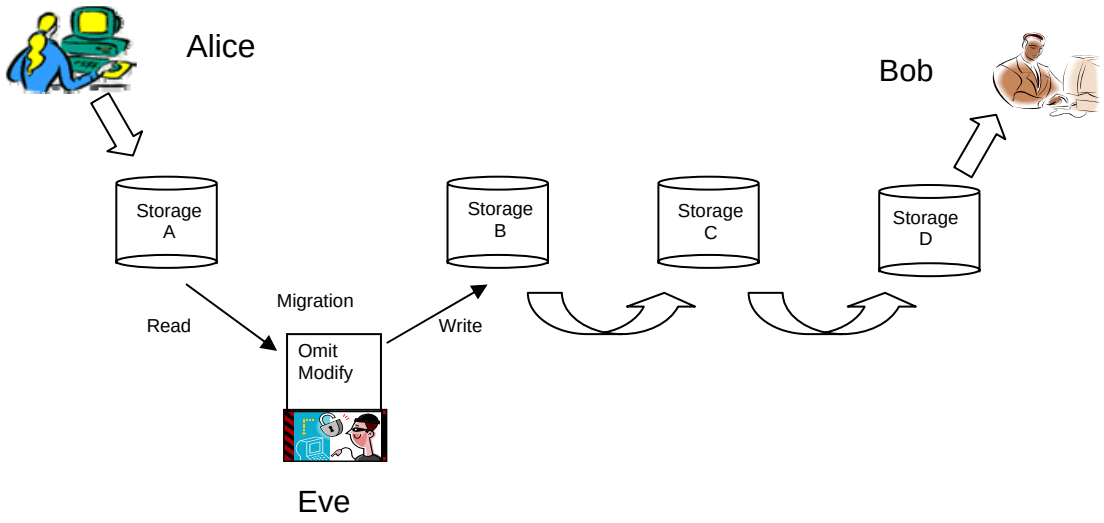
Some compliance regulations like HIPPA require companies to keep all medical records in encrypted format for preventing the privacy of data. Most businesses also maintain other critical records, like financial statements in an encrypted form to prevent them from being accessed by users who do not have the adequate_privilege. Unfortunately encryption schemes often get compromised: Keys are leaked, the underlying encryption algorithms are broken and so on. The risks are even higher in this case because of the very large timelines over which typical records have to be maintained. The only way to prevent critical data from being exposed when such a thing happens is to re-encrypt the entire data with a new key and delete the old data. If the data is stored on a Write-Once media, the data must be migrated to a new device by re-encrypting it, while the original volume must be taken offline to prevent data from being leaked.

4.1.2 Changes in corporate structure

Another common secnario requiring data migration is with changes in corporate structure. For example when a company splits, merges or gets acquired, the ownership and liability of data changes. In such cases, data is either consolidated (in case of company merges) or split (for company splits).

4.2 Current Approaches are Insecure

The following figure illustrates a typical migration scenario: Some user Alice commits a document on Storage A. The data is migrated by Eve through multiple devices B, C and D. Finally Bob reads the data from D. If Eve is malicious, she can delete/edit records during migration. This is particularly a threat if Eve is a company insider, say one of the C* people and can control and influence the migration process.



Traditionally migration is carried out by directly copying the data from one device to another. This however is not secure from insider attack: Eve has complete control over the migration process and hence can omit any records during migration.

One possible approach can be to build the migration support into the WORM device. The migration running code inside the WORM won't be accessible to Eve and hence cannot be compromised. Such on device migration is currently available on storage servers too, for example Network Appliances SnapMirror technology, which supports device to device data mirroring. Unfortunately just securing the migration code is not adequate. Eve can compromise the link between the source and destination WORM devices. It is still possible to address this problem by running a secure migration protocol between the source and destination device to ensure that all the files have been completely and correctly copied. At the end of the secure protocol, the source device can generate a certificate which can be verified by Bob.

There are however more subtle problems with this approach. Firstly, two devices must be compatible to communicate and execute the migration protocol. This has many practical implicationis: Both the devices must have the similar physical interface (fibre channel/ ethernet etc), the same file access protocol (NFS,CISF etc) and must support the same communication

protocol. This would typically force businesses to be locked into devices from a specific vendor, which is often not desirable. The second issue is with the complexity of the migration process itself - Migration is usally more than making an exact replica: Data is re-encrypted, "unimportant" files are left out, directory tree is rearranged, only a subset of the directory tree is copied (for example while retrieving from backup), data from multiple sources are consolidated and so on. Migration hence is usually carried out by administrators by writing customized migration scripts and with substantial human involvement. It is almost impossible to build a generic migration framework which can support all possible migration scenarios and build into the WORM device.

4.3 Possible Direction of Research

We are currently developing techniques for supporting trustworthy data-migration. Specifically we are exploring if it is possible to enhance the storage device with some additional feature to support trustworthy migration. One specific idea is to add some secure-coprocessor to the storage device which can accept any auditing code, execute the code and return a certificate of execution. Every migration operation can be accompanied by downloading an auditing code in the WORM storage and recording the certificate. The certificate testifies to the auditing code used, the files residing on the server, the associated directory structure, the changes to the files and directories that should take place during migration, the reasons for those changes, and the trustworthiness of previous migrations. We are currently developing on

References

[1] The Enterprise Storage Group, Inc. Compliance: The effect on information management and the storage industry, May 2003.

[2] Congress of the United States of America. Sarbanes-Oxley Act of 2002, 2002. Available at http://thomas.loc.gov.

[3] EMC Corp. EMC Centera Content Addressed Storage System, 2003. Available at http://www.emc.com/products/systems/centera.jsp.

[4] M. F. Fontoura, A. Neumann, S. Rajagopalan, E. Shekita, and J. Zien. High performance index build algorithms for intranet search engines. In VLDB'
2004, 2004.
[5] B. A. Huberman, P. L. T. Pirolli, J. E. Pitkow, and R. M. Lukose. Strong regularities in World Wide Web surfing. Science, 280(5360):95{97, 1998.

[6] IBM Corp. IBM TotalStorage DR550, 2004. Available at http://www-1.ibm.com/servers/storage/disk/dr.

[7] Network Appliance, Inc. SnapLockTM Compliance and SnapLock Enterprise Software, 2003. Available at
http://www.netapp.com/products/filler/snaplock.html

[8] Securities and Exchange Commission. SEC Interpretation: Commission Guidance to Broker-Dealers on the Use of Electronic Storage Media under the Electronic Signatures in Global and National Commerce Act of 2000 with Respect to Rule 17a-4(f), 2001. Available at http://www.sec.gov/rules/interp/34-44238.htm.

[9] Socha Consulting LLC. The 2004 Socha-Gelbmann Electronic Discovery Survey, 2004.

[10] Sony Corp. AIT-2/AIT-3 WORM Drives & Libraries, 2003. Available at http://www.storagebysony.com/products/prod_hilite4.asp.

[11] The Enterprise Storage Group, Inc. Compliance: The effeect on information management and the storage industry, May 2003.

[12] A storage solution case study http://www.netapp.com/library/cs/westbend.pdf

Appendix: SEC 17a-4

The following are from SEC 17a-4. These stipulate that all records must be immediately committed on a Write-Once media.

"*The records required to be maintained and preserved pursuant to Secs. 240.17a-3 and 240.17a-4 may be immediately produced or reproduced on ``micrographic media'' (as defined in this section) or by means of ``electronic storage media'' (as defined in this section) that meet the conditions set forth in this paragraph and be maintained and preserved for the required time in that form."*

The electronic storage media referred has been defined later as

(ii) The electronic storage media must:

1. *Preserve the records exclusively in a non-rewriteable, non- erasable format;*
2. *Verify automatically the quality and accuracy of the storage media recording process;*
3. *Serialize the original and, if applicable, duplicate units of storage media, and time-date for the required period of retention the information placed on such electronic storage media; and*
4. *Have the capacity to readily download indexes and records preserved on the electronic storage media to any medium acceptable under this paragraph (f) as required by the Commission or the self-regulatory organizations of which the member, broker, or dealer is a member.*