Chapter 23

INFERRING SOURCES OF LEAKS IN DOCUMENT MANAGEMENT SYSTEMS

Madhusudhanan Chandrasekaran, Vidyaraman Sankaranarayanan and Shambhu Upadhyaya

Abstract A document management system (DMS) provides for secure operations on a distributed repository of digital documents. This paper presents a two-phase approach to address the problem of locating the sources of information leaks in a DMS. The initial monitoring phase treats user interactions in a DMS as a series of transactions, each involving content manipulation by a user; in addition to standard audit logging, relevant contextual information and user-related metrics for transactions are recorded. In the detection phase, leaked information is correlated with the existing document repository and context information to identify the sources of leaks. The monitoring and detecting phases are incorporated in a forensic extension module (FEM) to a DMS to combat the insider threat.

Keywords: Document management system, insider threat, information leaks

1. Introduction

Digital documents have become the principal vehicle through which organizational information such as email, public memos and proprietary information are created and shared. Initially, digital documents were shared using portable media (floppy disks). Eventually, loosely-structured networked collaborations were created where documents were emailed in order to share information. However, as the value of the exchanged information grew, so did the security threats. Prompted by the increasing threat level and the importance of document content, sophisticated document management systems (DMSs) were developed to automate and secure document creation, check-in and check-out processes. Authentica [4] and Microsoft Information Rights Management [9]

Please use the following format when citing this chapter:

Chandrasekaran, M., Sankaranarayanan, V. and Upadhyaya, S., 2008, in IFIP International Federation for Information Processing, Volume 285; *Advances in Digital Forensics IV*; Indrajit Ray, Sujeet Shenoi; (Boston: Springer), pp. 291–306.

are examples of DMSs. These systems protect the documents from external threats, e.g., by automatically encrypting every document. Thus, even if the file server containing the documents is compromised, information in the documents is not compromised. While numerous security mechanisms have been designed to safeguard documents from external intruders, DMSs and the documents they contain remain vulnerable to insider attacks.

This paper describes a two-phase extension to an existing DMS, called the forensic extension module (FEM), that identifies the sources of information leaks – a common form of insider abuse. During the first phase, every user action is logged by a monitoring component. In the second phase, when an information leak is discovered, audit data along with data gathered during the monitoring phase are used to attribute the sources of the leak. As a proof of concept, the FEM is implemented as an add-on to Word 2003, where it is seamlessly integrated into the process flow and conducts evaluations in a virtual environment.

2. Related Work

Incidents of information leaks involving Microsoft Word documents have led to the development of several add-ons for document editors to combat the threat. For example, Microsoft's Remove Hidden Data tool [8] removes all meta tags, field codes and revision information from Word documents. Microsoft's Word Redaction tool [7] is an add-on that redacts information from Word 2003 documents. Note that these add-ons were not created for forensic purposes; they are merely filters that prevent unintentional information leaks from documents that have been declared fit for public consumption. In the context of our work, these tools are important for two reasons: (i) they expose and cleanse information in documents that is not immediately visible to end users, and (ii) they may leak information and, therefore, should be considered when performing forensic analysis.

Recent efforts in document forensics have focused on several issues, including document reconstruction from deleted fragments [15], retrieval of hidden documents via file system analysis [3], detection of masquerades for the purpose of document access [13], and mitigation of illicit system and log file tampering [14]. Although these techniques can enhance the overall detection capabilities in a DMS, they do not specially address the problem of information leaks, which is the crux of our research. In fact, our focus is on "information leak forensics" in a DMS rather than traditional "document forensics."

3. Document Management Systems

A document management system (DMS) is a repository of digital documents that provides the functionality for shared editing, collaboration, check-in and check-out, and various security features. The predominant security features are document encryption and custom security policy settings. Note that the DMS security features are document-format-specific and viewer-specific as opposed to file-system-specific. In a typical DMS, users interact with a secure document editor like Microsoft Word 2003 or Adobe Acrobat. The editor is responsible for authenticating users, communicating with the file server, retrieving documents and enforcing custom security policies on the documents. The security policies are mostly static policies that dictate user rights to documents. Read, edit and print permissions may be assigned for documents. Additional fine-grained policies may be set for specialized document formats.

Each document in a DMS is assigned a type and classification. The document type usually indicates the nature of the information content (e.g., Financial, News, Technology). The classification, on the other hand, indicates the sensitivity of the document (e.g., Top Secret, Secret, Classified, Unclassified, Declassified, Public). In addition, user roles (e.g., Secretary, Software Developer, Project Manager, Board Member, Chief Executive Officer) are usually defined based on the organization's functions. Security policies are defined based on user roles and document classifications and types. In some DMS architectures, documents may also be watermarked or digitally signed to establish the authenticity of their content, proof of ownership and non-repudiable statements pertaining to access histories.

3.1 Characterizing Information Leaks

Modern document formats, most notably the format used by Microsoft Word, have provisions for storing data in various sections that may not be immediately visible to users:

- **Actual Document Content:** This section, which contains text, pictures, embedded objects and comments, is usually what the author and the readers view.

- **Document Metadata:** This information, e.g., document author, time of last edit and time of last printing, is not immediately visible to users. However, the metadata can unintentionally reveal confidential information such as author name and document classification.

- **Content Change/Revision Information:** Modern document editors have a provision for tracking changes to documents. This is usually the starting point for shared document editing, where a document is marked with "Track Changes." All changes are recorded, but the final document may not present the earlier versions. However, the original content and all the revisions are embedded in the document unless they are explicitly removed.

- **Content Versions:** This feature enables a single document to store multiple versions of the same document over time.

Information leaks involve the release of these different types of information contained in documents. Depending on the nature of the leak, the revealed information can cause considerable damage to the concerned organization.

3.2 Protection from Inadvertent Leaks

Document management systems are designed to enable collaborative document editing by ensuring workflow integration and incorporating security mechanisms where necessary. The dual goals of workflow integration and security control are at the root of any information leak.

Consider a scenario where a document D1 has type Financial and is classified as Secret. A user with the role of Accountant is authorized to work on the document with read and edit permissions. At the end of the financial year, the user transfers some summary information from D1 to a public document D2 for a press release. Since the user has read and edit permissions on D1, he can perform this information transfer, which involves the use of copy and paste commands using the document editor. This scenario illustrates two important characteristics of a DMS:

- The DMS must permit information transfer from document D1 to a public document. This is in accordance with the organization's workflow requirements (i.e., non-interference).

- The DMS must protect D1 by ensuring that only authorized users are allowed to access it. In accordance with the prevailing security policies, it must prevent the Accountant from executing a print operation on document D1.

Many commercial entities tout tight integration with workflow (and non-interference) as a feature of their DMS products. However, it is trivial to observe that the Accountant is in a position to transfer information from document D1. While security policies can be implemented to pre-

Table 1. Notation used for modeling leaks.

U	Set of DMS users: $u_i \in U$
D	Set of DMS documents: $d_i \in D$
C	Set of DMS document classifications: $c_i \in C$
S_D	Set of DMS document security identifiers (DSIDs): $s_d \in S_D$ such that $\forall d \in D, \exists s_d \in S_D$ specifying the corresponding security level
S_U	Set of DMS user security identifiers (SIDs): $s_u \in S_U$ such that $\forall u \in U, \exists s_u \in S_U$ specifying the corresponding security level
$u_i \rightarrow d_j$	User u_i accesses document d_j
d_{open}	Leaked document is found in the open

vent this information transfer [11], it is, nevertheless, possible for a malicious insider to leak information within the purview of normal workflow processes. On the other hand, a security policy that completely prevents such information flow would interfere with normal workflow processes.

An information leak occurs when information at a higher classification level becomes available at a lower level. Information leaks may be categorized as inadvertent or premeditated. Any undesirable information transfer occurring as part of a legitimate workflow process is termed as inadvertent. For example, if the Accountant wishes to create a new public document D2 with the same formatting as D1, he may initiate a file copy of D1 to D2, which replaces the original content of D2. Then, he proceeds to edit the content and create the public document D2. However, documents D1 and D2 have the same metadata (author information, original creation time, last printed time, etc.). Absent DMS detection and mitigation functionality, this metadata is leaked when document D2 is published on the corporate website.

4. Modeling Information Leaks

This section formally characterizes information leaks. Table 1 presents the notation used for modeling leaks.

Definition 1 *Information Potential (IP): The information potential of a document $d_i \in D$ is defined by:*

$$IP(d_i) = (s_{d_i} \sum s_{u_j}) / \sum u_j \qquad \forall u_j \in U : u_j \rightarrow d_i.$$

The information potential of a document expresses the importance of the information it contains. Note that the information potential of a document can be trivially defined to be its DSID. The SID of a document is generally assigned a value in the range [0,1] depending on its

criticality; Public documents are assigned a value of zero while Top Secret documents are assigned a value of one. The relative importance of a document is expressed by including a weighting for the levels of the users who access the document. Thus, a document is accorded importance based on its SID as well as on the levels of the users who access it [5].

Definition 2 *Document Similarity Set (D_{sim}): The document similarity set D_{sim}, corresponding to a document d_{open} found in the open, is a set of documents that have contributed to d_{open} along with their respective similarity scores. Thus, D_{sim} is a set of tuples of the form <document, score> defined by:*

$$D_{sim}(d_{open}) = < d_1, sc_1 >, < d_2, sc_2 >, \ldots, < d_k, sc_k > \text{ where } d_k \in D \text{ and } sc_1 \geq sc_2 \geq \ldots \geq sc_k.$$

Definition 3 *Information Leakage Value (IL_{val}): The value of the information leaked in d_{open} is defined by:*

$$IL_{val}(d_{open}) = \sum_{i=1}^{|D_{sim}|} IP(d_i) \times sc_i.$$

IL_{val} represents the information similarity measure for the document d_{open} with respect to the documents contained in D_{sim}. This definition depends only on the information potential of documents, not on the quantity of information transferred. Thus, a single word transferred from d_1 to d_2 is equivalent to the transfer of a sentence or paragraph or section. Since we do not have a mechanism to detect the importance of the content of a document, this definition is the best we can use to quantify information transfer in a DMS.

4.1 Problem Definition

Given a document d_{open}, constructed by the complete or partial composition of one or more documents in $D = d_1, d_2, \ldots, d_n$, that constitutes an information leak, return a list of suspects, i.e., users $U_{suspects} = u_1, \ldots, u_k : u_i \in U$. Thus, the goal is to deduce the list of suspects, possibly a single user, whose actions resulted in the information leak.

The document d_{open} could be a confidential piece of information leaked intentionally and discovered by a network trace or by examining log files. The insider could create d_{open} as a composition of the documents to which he has access. Thus, d_{open} may contain some confidential infor-

mation and mostly public information; the idea being to mix information so that the leak is not detected and traced to the insider.

It is also possible that an information leak could be inadvertent, i.e., the information transfer was intended for legitimate purposes but was later found to be in violation of the security policy. Generally, we assume that $d_{open} = d_1 \circ d_2 \circ \ldots \circ d_k$, where $d_1, d_2, \ldots, d_k \in D$, i.e., d_{open} is the composition of documents d_1 through d_k. For simplicity, we assume that d_{open} only contains information from $d_1, d_2, \ldots, d_k \in D$, although the insider might add other commonly available information to create document d_{open}. However, as discussed below, adding spurious information does not impact the detection of information leaks.

4.2 FEM Algorithm Preliminaries

To attribute the source of an information leak, it is essential to capture all the changes made to documents in a DMS, preferably in a succinct way. We use rooted, labeled trees to model the transmutations that documents undergo in their lifetime. We denote the tree as $T = (D, E, \epsilon)$ where D is a set of nodes representing different versions of the documents after edit sessions. Node $r \in D$ is a special node that forms the root of the tree.

All the documents contained in a DMS at any point in time correspond to nodes that are connected directly to the root r. $E \subseteq D \times D$ is the set of edges in the tree. An edge $(d_{i_j}, d_{i_k}) \in E$ denotes the transition that a document d_i undergoes as a result of edit operations. In other words, d_{i_k} is a version of the document that has evolved from its previous version d_{i_j}. The sequence $\epsilon = e_1, e_2, e_3, \ldots, e_k$ is the edit script that transforms the document from one version d_{i_j} to another d_{i_k}. In turn, each e_i represents an edit operation that is applied as a part of an edit script ϵ. L is a set of version labels. Each version of the document $d_i \in D$ after a successful edit operation is uniquely identified with a different name d_{i_l} corresponding to its label l (version).

Any sensitive information that could have leaked from a document with a higher classification to a document with a lower classification is encapsulated by the edit script. We define the set of edit operations [2] that can be applied in an edit script ϵ_1 to transform document version d_i to document version d_j (i.e., $d_i \rightarrow^{\epsilon_1} d_j$).

- **Insertion:** Each DMS document is considered to be a flat file represented as a $p \times q$ rectangular grid where p is the column width of the document and q is the number of lines in the document. Each $p \times q$ point in the grid is mapped to a single ASCII character. The insert operation $INS(p_1, q_1, content)$ inserts the ASCII characters

dictated by the content beginning at (p_1, q_1) in the grid. The content previously located at (p_1, q_1) and beyond is shifted and concatenated with the inserted content.

- **Deletion:** Deletion is the inverse of insertion. $DEL(p_1, q_1, p_2, q_2)$ truncates the content located between $(p_1,\ q_1)$ and $(p_2,\ q_2)$.

- **Update:** The $UPD(p_1, q_1, p_2, q_2, content)$ operation replaces the content between $(p_1,\ q_1)$ and $(p_2,\ q_2)$ with the specified content. An update operation is equivalent to successively applying the $DEL(p_1, q_1, p_2, q_2)$ and $INS(p_1, q_1, content)$ operations.

- **Copy:** $CPY(p, q, d_s)$ is a special operation that does not change the content of document d_s. It is used to capture "content highlighting" and "copy to clipboard" events, which occur when information is transferred within a document or between documents.

- **Glue:** $GLU(p_1, q_1, p_2, q_2, content, d_s)$ is similar to the update operation. The only difference is that the content between (p_1, q_1) and (p_2, q_2) is replaced with the content copied from the clipboard taken from document d_s using the CPY operation. If $(p_1,\ q_1)$ is equal to (p_2, q_2), the GLU operation becomes equivalent to $INS(p_1, q_1, content)$. It is possible that content is manually transferred from document d_u to d_v. Such an information transfer is recorded as an INS operation instead of the CPY and GLU operations.

4.3 FEM Trace-Back Algorithm

This section describes the algorithm for tracing the sources ($U_{suspects}$) of an information leak.

First, the leaked information is correlated with a set of documents D_{sim} in the DMS. Next, it is determined if the leak is caused by CPY and GLU operations in an edit script ϵ that transfer information from a document d_k with a higher classification to a document d_i with a lower classification to produce a new version d_j. In such an instance, the content pasted from the clipboard by the GLU operation is checked to see if it matches the leaked content.

An information leak can also take place across multiple edit sessions spanning multiple documents. This is similar to a "slow poisoning" attack where D_{sim} only matches the final version of the document, say d_j, even though the information leak could have occurred in part during previous editing sessions.

Algorithm 1 Malicious Insider Detection Algorithm

Require: IL_{tsh} and audit logs as specified in the monitoring phase
Ensure: Output of $U_{suspects}$

1: Evaluate $D_{sim}(d_{open}) = < d_1, sc_1 >, < d_2, sc_2 >, \ldots, < d_k, sc_k >$
2: Calculate $D_{high} = \ d_i \in D_{sim} : IL_{val} > IL_{tsh}$
   Calculate $D_{low} = \ d_i \in D_{sim} : IL_{val} \leq IL_{tsh}$
3: Calculate $U_{high} = \ u_i \in U : \forall d_j \in D_{high}, u_i \rightarrow d_j$
   Calculate $U_{low} = \ u_i \in U : \forall d_j \in D_{low}, u_i \rightarrow d_j$
4: **for all** d_i in D_{sim} **do**
5: $\quad < d_{i1}, d_{i2}, \ldots, d_{ik} >$ = Greedy-Collate(GLU/CPY)
6: $\quad U_{suspects}$ += $u : u_l \rightarrow d_{ij}$ where $d_{ij} \rightarrow^{\epsilon} d_{ik}$, $\forall u_l \in U, \forall d_{ik} \in D_{sim}$
7: **end for**

The algorithm collates the contents of GLU operations from edit scripts from previous sessions in a greedy manner to check if the collated content matches the leaked content. All the users who initiated the edit script are deemed as suspects. However, not all information leaks occur due to CPY and GLU operations. For example, a user might try to reproduce a document by reading it and typing its content ("content jacking"). The algorithm only considers the content involved in the INS and UPD operations and generates an information graph [11] to determine the documents in D_{sim} that were opened concurrently with the public document d_j that contains the leaked information.

Algorithm 1 presents the steps involved when only GLU operations are considered. The sub-procedure Greedy-Collate is self explanatory.

The algorithm is easily extended to incorporate content collation implemented by the INS and UPD operations. The algorithm also (optionally) takes as input an information leak threshold value (IL_{tsh}) beyond which any information transfer is considered to be a leak. If IL_{tsh} is specified, then D_{sim} in Step 4 can be replaced by D_{high} and U in Step 6 can be replaced by U_{high}.

5. Forensic Extension Module

The forensic extension module (FEM) has two phases, monitoring and analysis. The monitoring phase is an online process, i.e., it takes place whenever there is user activity. The components used in this phase augment standard DMS auditing procedures. Each user action/interaction with the documents in the repository is recorded and relevant context information is collected. Along with each activity, information about the document classification, time of transfer, etc. is also logged. Various metrics are computed from the logs to understand the specific actions

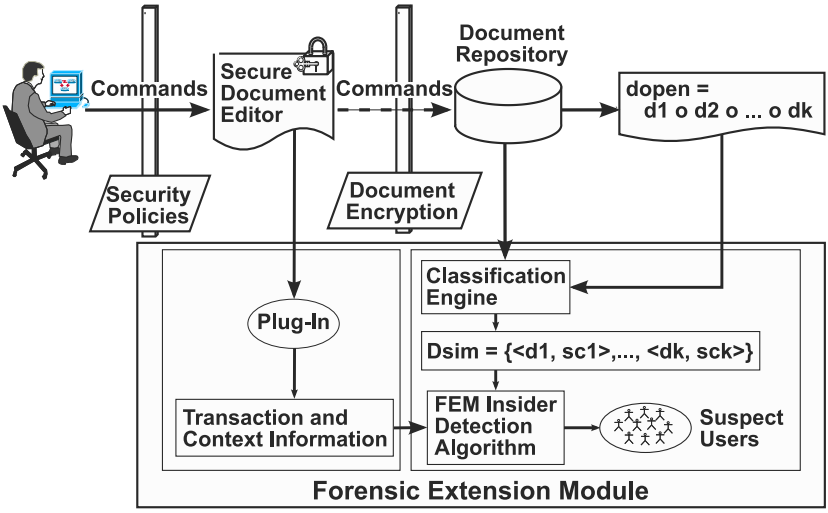

Figure 1. Incorporation of a FEM in a DMS.

taken by users and their intent. For example, a flurry of document accesses (reads and searches) may indicate exploratory activities that are not part of the workflow process. A simple query submitted to the logs can provide all the instances of these document accesses. Similarly, transaction time and transaction origin (within the organizational perimeter or external access via a VPN connection) are indicators of surreptitious activity.

The second FEM phase, which involves offline analysis, is initiated whenever a leaked document is found in the open. The leaked document is first correlated with the document repository. Based on the correlation, the set D_{sim} is constructed (as in Definition 3). Finally, the DMS audit logs and the metrics computed during the monitoring phase are used to obtain the list of suspects $U_{suspects}$.

5.1 Integration Issues

Microsoft Word 2003 was chosen as the DMS hosting platform. It (i.e., Office 2003) comes with a digital rights management feature (Information Rights Management), which enables document protection, including the specification and enforcement of custom policies. Most of the Word Object Model interfaces are exposed as standard SDKs, which facilitate the addition of custom plug-ins (called "add-ins" in Microsoft's documentation). Thus, the FEM was implemented as an add-in to Word 2003 and integrated with the DMS process flow as shown in Figure 1.

5.2 Implementation Issues

During the monitoring phase, the FEM is supposed to log information on all user transactions involving DMS documents. However, this is difficult to implement because document editors such as Microsoft Word and OpenOffice were designed primarily as document editing tools rather than inhibition tools. Over the years, they have evolved from using simple text documents with formatting tags to supporting complex documents with embedded software that provides advanced features. Thus, a Microsoft Word document is specified as an XML schema with tags and binary-encoded streams for various portions of the document. A side effect of these tags is that, from a document editing viewpoint, they do not form part of the content, but can conveniently be used to transfer critical information. The interface exposed by the Word Object Model enables notification for certain document operations (e.g., opening and printing) but does not provide a hook for logging information transfer. For example, if a user were to copy and paste information from one document to another, no API is available to hook the copy and paste events. To overcome this limitation, information required during the monitoring phase is obtained by enabling the "Track Changes" feature for every document edited by a user and creating a log of all the files modified in a session.

6. Evaluation

The following components were used to emulate a DMS in order to evaluate the effectiveness of the FEM trace-back algorithm.

- **Document Corpus:** The corpus contained Microsoft Word 2003 documents created using the 20 newsgroup data set [12]. The documents were classified into five categories: Top Secret, Secret, Confidential, Classified and Public. The twenty newsgroups in the data set were uniformly distributed among the five document classifications. For example, the posts (in plain text format) in `talk.politics.guns` were converted to Microsoft Word documents and were uniformly distributed under the five classifications. Note that converting the posts to Word document format is slightly more involved than merely renaming the files with a `.doc` extension. A simple helper tool that instantiates a Word Application Object and seamlessly converts plain text files in a given directory to Microsoft Word 2003 documents was used for this purpose. This tool performs the equivalent of manually renaming the newsgroup post with a `.doc` extension, opening the renamed document

Table 2. Access control matrix.

	Admin	**Manager**	**Pgmmr**	**Intern**	**Contr**	**Secy**
Top Secret	r, w, p	r, w, p	-	-	-	-
Secret	r, w, p	r, w, p	r	-	-	-
Confidential	r, w, p	r, w, p	r, w	r	-	-
Classified	r, w, p	r, w, p	r, w, p	r, w	r, w	-
Public	r, w, p	r, w, p	r, w, p	r, w, p	r, w	r, w, p

in Word, accepting the default encoding, and saving the file in the latest Word format.

- **User Set:** Five classes of users were defined: Administrator $\succ$ Manager $\succ$ Programmer $\succ$ Intern $\succ$ Contractor $\succ$ Secretary. Note that these roles do not naturally produce a linear hierarchy; this hierarchy was chosen only for the proof-of-concept implementation. A fixed access control matrix was used to specify the rights (r: read, w: write and p: print) possessed by the five classes of users to the five document categories (Table 2).

- **FEM:** The FEM add-in for Microsoft Word 2003 provides audit logging and forensic capabilities. It was implemented in C# using Visual Studio 2005 under .NET Framework 2.0 and Windows Vista. When installed, the add-in places a "FEM toolbar" in Word.

  The FEM toolbar has three controls:

  - A drop-down combo box titled "Role Choice," which enables the user to choose a role (e.g., Intern) for the particular session. Based on the role, access control policies are applied that allow a test subject to open documents with the classifications dictated by the access control matrix in Table 2.

  - A button titled "Start FEM Logging," which is used to initiate logging after a role is chosen. Note that the logging is not performed by a third-party process, but by Microsoft Word, whose functionality is extended by the FEM add-in. Thus, the FEM add-in is truly an extension to the DMS (where Microsoft Word serves as the document viewer and editor). The FEM add-in can be extended to dump the logs to a remote database or a server (e.g., Windows 2003 Server).

– A "Send Logs" button is available to (optionally) send the log by email upon completion of a session. This feature can be eliminated if a trusted DMS platform is used.

The FEM evaluation experiment gathered data from thirteen virtual users, each with an assigned role. The users were asked to transfer information from the highest classification level to which they had access (based on the access control matrix in Table 2) to a Public document. They were allowed to browse the Internet and copy and paste information from the Internet to obfuscate detection attempts. Note that because the users only performed the operations defined by the access control matrix, a simplified version of the detection algorithm was applied where the content derived from the "Track Changes" feature was merged with the log file to infer the actions performed by users that led to information leaks.

Document similarity was tested using a simplified "diff" algorithm [10] (other algorithms such as the Term Frequency Inverse Document Frequency (TF-IDF) algorithm [6] can also be used). Detection would have been much easier if the documents had been structured (e.g., using XML [1]).

Two of the thirteen users chose to not initiate any information leaks. Nine users who leaked a significant amount of information were detected by the FEM. Using the check-in and check-out feature of the DMS and the difference between the two versions of the documents, it was relatively straightforward to identify the nine users as final suspects. Two users who leaked information were not detected at all. One user transferred a single number from a Top Secret document. As this change was minute, the transfer did not score a high enough similarity score for the algorithm to investigate the user. The other user changed the content of a Public document without actually copying information directly from a Secret document. Furthermore, the content of the destination Public document was scattered throughout the document making it difficult for the FEM (or any computer program) to analyze the content.

The performance of the FEM was very reasonable; however, it has inherent limitations that stem from the nature of the problem and the myriad possibilities that exist for information leaks. Users cannot have their workflow affected. In a DMS, this translates to unhindered information flow subject to static and context-specific security policies. If an information leak occurs, the FEM can narrow the list of suspects and, in favorable circumstances, can identify the single malicious insider. The FEM relies on the similarity between the leaked document and the document repository; this is correlated with the information transfer ini-

tiated by users. However, a major limitation arises from the fact that the similarity score is computed for raw text while information transfer may (also) be in terms of pictures, WordArt, AutoShapes, or even custom embedded objects. These types of information transfer can be regarded as steganographic in nature; they are simple for a human to perform but very difficult for a computing system to recognize or categorize. Although this problem has some resemblance to the (reverse) Turing test [16], its scope is much larger. Indeed, the lack of computing approaches (possibly based on artificial intelligence) for recognizing such "information" significantly impacts the efficacy of a FEM-like approach for detecting steganographic forms of information transfer.

7. Conclusions

Information leaks in DMSs are a major security threat, but little work has been done on detecting and mitigating them. Current approaches, which are effective at detecting infractions when information is transferred between documents of different classifications, are impractical for two reasons. First, defense mechanisms are intrusive and can significantly hinder workflow processes. Second, loosely-framed DMS policies may permit actions that result in information leaks. Our FEM solution addresses these two issues using an information leak metric and coupling it with audit data collected during the monitoring phase. Its proof-of-concept implementation as an add-in to Microsoft Word 2003 is likely the first time that forensic functionality is integrated in an environment where a threat vector (malicious insider) is not addressed. Our future research will investigate extensions to the FEM framework for detecting and mitigating information leaks propagated through other mechanisms such as printed materials, steganography and human channels.

References

[1] S. Chawathe and H. Garcia-Molina, Meaningful change detection in structured data, *ACM SIGMOD Record*, vol. 26(2), pp. 26–37, 1997.

[2] S. Chawathe, A. Rajaraman, H. Garcia-Molina and J. Widom, Change detection in hierarchically structured information, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 493–504, 1996.

[3] K. Eckstein and M. Jahnke, Data hiding in journaling file systems, *Proceedings of the Fifth Annual Digital Forensics Research Workshop*, 2005.

[4] EMC Corporation, Authentica Software, Hopkinton, Massachusetts (software.emc.com/microsites/regional/authentica).

[5] A. Garg, S. Pramanik, V. Shankaranarayanan and S. Upadhyaya, Dynamic document reclassification for preventing insider abuse, *Proceedings of the Fifth Annual IEEE SMC Information Assurance Workshop*, pp. 218–225, 2004.

[6] D. Grossman and O. Frieder, *Information Retrieval: Algorithms and Heuristics*, Springer, Dordrecht, The Netherlands, 2004.

[7] Microsoft Corporation, Office 2003 Add-In: Word Redactionv1.2, Redmond, Washington (www.microsoft.com/downloads/details.aspx?FamilyID=028c0fd7-67c2-4b51-8e87-65cc9f30f2ed&displaylang=en).

[8] Microsoft Corporation, Office 2003/XP Add-In: Remove Hidden Data, Redmond, Washington (www.microsoft.com/downloads/details.aspx?FamilyId=144E54ED-D43E-42CA-BC7B-5446D34E5360&displaylang=en).

[9] A. Mehta, Office Space: Information rights management in Office 2003, TechNet, Microsoft Corporation, Redmond, Washington (technet.microsoft.com/en-us/magazine/cc160822.aspx), 2003.

[10] E. Myers, An O(ND) difference algorithm and its variations, *Algorithmica*, vol. 1(2), pp. 251–266, 1986.

[11] S. Pramanik, V. Sankaranarayanan and S. Upadhyaya, Security policies to mitigate insider threats in the document control domain, *Proceedings of the Twentieth Annual Computer Security Applications Conference*, pp. 304–313, 2004.

[12] J. Rennie, 20 Newsgroups (people.csail.mit.edu/jrennie/20Newsgroups).

[13] V. Sankaranarayanan, S. Pramanik and S. Upadhyaya, Detecting masquerading users in a document management system, *Proceedings of the IEEE International Conference on Communications*, pp. 2296–2301, 2006.

[14] B. Schneier and J. Kelsey, Secure audit logs to support computer forensics, *ACM Transactions on Information and System Security*, vol. 2(2), pp. 159–176, 1999.

[15] K. Shanmugasundaram and N. Memon, Automatic reassembly of document fragments via context based statistical models, *Proceedings of the Nineteenth Annual Computer Security Applications Conference*, pp. 152–159, 2003.

[16] L. von Ahn, M. Blum and J. Langford, Telling humans and computers apart automatically, *Communications of the ACM*, vol.47(2), pp. 56–60, 2004.

[17] W. Wang and T. Daniels, Building evidence graphs for network forensics analysis, *Proceedings of the Twenty-First Annual Computer Security Applications Conference*, pp. 254–266, 2005.