# Automated extraction of behavioural profiles from document usage

**S D Wolthusen**

*Both human analysts and particularly automated tool suites are capable of deriving sensitive information and conclusions from collections of data items that individually cannot be considered critical or sensitive. This activity of analysing and correlating material that is not immediately related is, in fact, highly desirable in many application areas and cannot be controlled precisely in advance. The decision whether a program or an analyst is performing searches and correlations beyond the scope of his authorisation or current mission can frequently be determined only ex post based on a heuristic analysis of documents accessed.*

*In this paper we describe a mechanism for the instrumentation of operating systems to obtain information on the documents and resources accessed by arbitrary processes. Such a mechanism could be an important component of the infrastructure of an operational risk management system, generating an audit trail for compliance and forensic investigation, and acting as a sensor generating data for analysis. Addressing the latter application, the paper also outlines an approach for extracting textual information and metadata from accessed documents, regardless of the application program and workflow mechanisms used, without unduly impeding either workflows or operator performance.*

*This information can then be subjected to an heuristic analysis based on natural language processing to extract the semantic context of each document or segment. Clustering this content and extracting the conceptual patterns that a user has accessed can then allow abnormal behaviour to be identified. This can then be refined further to determine heuristically whether the authorised remit of the user has been breached and whether an investigation is warranted. We argue that the risk of misbehaviour can be reduced while at the same time increasing productivity. This is made possible by enhancing the degree of freedom for individual users to act in the interest of their mission objectives and at the same time providing automated mechanisms for analysing user behaviour.*

## 1. Introduction

In environments where sensitive information must be processed in a way that requires a certain amount of flexibility (e.g. to ensure cross-fertilisation in intelligence analysis and also in research and development), choosing between highly restrictive security controls and providing staff with the access privileges required to conduct their work is a significant dilemma. Each course of action carries significant risks, although in the former course of action, these tend to be long term and harder to quantify since they are opportunity costs, i.e. discoveries and developments not made or productivity lost.

However, even when opting for a heavily controlled environment it must be observed that security controls, including mandatory controls, are limited in their granularity by several factors. A primary aspect governing these limitations is the complexity of the security policy to be transformed into access control rules. The identification of entities to which controls are to be applied is typically time-consuming and must take the dynamic creation and deletion of entities as well as changes in the security properties of individual entities into account. Even within the constraints imposed by the security models used, the full expressiveness of these controls can therefore not be used effectively owing to the overwhelming effort that would otherwise be required. However, another problem that is primarily of concern with unstructured data such as document collections is that the same controls do not allow the flexibility to designate selected portions of a given document or collection as having differentiated security properties. While it is possible to break up such collections into multiple components and assign security properties to each, this once again requires additional, often manual, effort and thereby further exacerbates the previously noted problem.

Finally, in many cases it is neither desirable nor possible to designate precisely the bounds imposed on the access of either an individual or, in some cases, an autonomous process to a document or document collection. This conclusion is motivated by two insights. Firstly, even though

individual documents and information may not be sensitive or classified as such, the process of collation itself and the integration of multiple such elements of information may result in a collection which a designating authority would classify as sensitive. Secondly, it is only through access to material for research and investigation that a number of legitimate and desirable insights can be generated. By constraining an individual's access unduly *a priori*, important insights may be lost.

In both of the cases described above, a decision on the legitimacy of a given line of enquiry can only be fully determined *ex post* by considering the collection process itself, the outcome, and the context in which this outcome was achieved. The context in turn consists of the identity of the entity conducting the enquiry, information on the status of the entity (e.g. association with an organisational unit or an ongoing project), and a history of this entity's past behaviour. Beyond areas with demonstrable security requirements (e.g. as required by law or other binding regulations), however, the formulation of detailed security policies and their technical implementation is limited in most organisations. As noted above, the cost of devising, co-ordinating, and implementing such policies, along with on-going maintenance efforts, appears to be outside an acceptable range and may well be counterproductive.

Based on earlier research [1], we argue that in such cases it will often be acceptable and desirable to employ only minimal (or otherwise feasible) security controls, but to augment these controls with an audit mechanism which in turn is coupled both with a behavioural anomaly detection mechanism and with forensic capabilities. For legitimate usage that is typically clustered around a selected number of tasks with limited deviations, this can direct the attention of security administrators to anomalous behaviour without requiring excessive initial and ongoing policy definition and implementation efforts. This provides a third option, in addition to the two outlined at the beginning of this section — the combination of coarse security controls, which can be established at reasonable cost, and an *ex post factum* mechanism which allows the partially automated and assisted validation of trust placed in staff.

It represents a shift in emphasis in risk management from prescriptive static policies to a more dynamic approach based on recognising high-risk operational situations and invoking the appropriate response.

The remainder of this paper is structured as follows — section 2 provides a brief outline of the architecture used to capture and analyse document access and usage patterns, while sections 3 and 4 discuss the instrumentation and data capture mechanisms. Section 5 contains a description of the reassembly and format extraction mechanisms, while section 6 briefly discusses the analytical mechanisms used to provide semantic clustering and anomaly detection capabilities. Finally, section 7 provides an overview of on-going and future research activities.

## 2. Architecture

The objective of the system architecture is to provide a fully transparent mechanism for observing any access to file systems and network data traffic that cannot be bypassed by application programs and, at the same time, also does not require modification of user or application program behaviour. The data to be captured by this interception mechanism consists of both metadata (e.g. file names and locations, timestamps, and access modes in the case of files, and source and destination addresses for network traffic) and the actual data retrieved. This data has many uses in an operational risk management (ORM) system, and the proposed mechanism is potentially a valuable component of ORM infrastructure.

The system is divided into four components:

- a suite of sensor components embedded in the host operating system that can intercept and monitor all data retrieval from all file systems (both local and via network file systems) and selected network protocols (primarily HTTP and S-HTTP over TLS),
- a sensor data extraction stack for analysing individual files and constrained data streams for known file and media types and, where possible, for extracting pertinent information into a normalised format for further processing by the anomaly detection and semantic modelling components,
- an anomaly detection mechanism operating both on the metadata level and the data extracted by the sensor data extraction mechanism,
- a semantic modelling module operating primarily on data extracted by the sensor data extraction mechanism to correlate and classify concepts analysed by a given entity and behavioural patterns exhibited in the process of analysis and data retrieval.

Since the raw data would prove quite voluminous if captured naïvely, the extent of interception must be limited based on the maximum acceptable degradation in response behaviour during operation, the storage and transmission capacity available for intercepted data, and the capabilities and performance of the mechanisms subsequently available for analysis. Particularly the latter trade-offs are, however, hard to assess quantitatively, since the need for further analysis or data to corroborate a given hypothesis will frequently arise only after the fact.

Moreover, the architecture described here assumes that the systems on which it is to be deployed are in a benign threat environment and administered by trusted personnel. This permits significant portions of analytical processing to

be performed on the monitored systems themselves since the total computational capabilities of even modest current personal computers significantly exceed those required by the user, except during brief periods of intense activity. As a result of this excess capacity, the system is designed to perform only the actual interception, pre-filtering, and storage of data synchronous with the operations of the user (or application program); the extraction, analysis, and reporting steps are deferred as necessary to limit the degradation in response time experienced by the user.

Moreover, only the interception, pre-filtering, and storage must be performed at the operating system kernel level; all other steps can be performed at the user level (albeit protected from all user processes), thereby significantly reducing the total developmental effort required. The requirement for a benign threat environment exists since otherwise tampering (e.g. initiating a hardware reset of the system) or outright destruction of the computer can erase or modify data required for analytical processes. While incidents such as power failures or software flaws can induce loss of data, the availability of additional computational power must be considered a significant factor in the trade-off.

## 3. File system instrumentation

An analysis of an entity's activities at the file system level requires that both the metadata associated with the operation and with the files themselves must be captured. This, however, would not yield sufficient insights into the actions performed by the entity as the actual content of the files may not be evident based on the available metadata.

It is therefore necessary to capture a file's content in addition to the metadata for subsequent analysis. Given that the strategies used by application programs for reading files on behalf of users differ considerably, information on the exact file locations accessed are typically not a reliable indicator of the information viewed and processed by the user. The granularity of analysis should therefore be constrained to individual files and their content, not to the access patterns exhibited in reading and writing these files. However, since the identification and subsequent extraction of textual data generally requires the entire file (see section 5) to be present for inspection, and analysis of the file also requires sufficient contextual data (see section 6), it is necessary to preserve the content of the entire file (or database record) regardless of which portion was actually accessed by the application process.

For advanced database management systems (e.g. the Oracle DRBMS family), recording an individual's access patterns to both individual records and, where necessary, record sets, is accomplished by inserting a trigger for this purpose. For read-only access, it is sufficient to note the precise table space, table row, and column (assuming a relational or object-relational DBMS), if the database metadata has been recorded earlier (i.e. when an entity logs into the given table space). Since database systems typically do not retain overwritten records, write access for overwriting and adding new records must be accompanied by a trigger which duplicates the new record in the audit trail. This overhead is clearly acceptable only in environments where read operations clearly dominate write accesses. However, given that the required accesses to the production data sets are read-only accesses without requirements for extensive locking, the performance impact is mainly owing to additional transactions rather than lock contention.

For file systems, no such mechanism for the generation of audit trails exists. To ensure transparency for all processes accessing file systems on a monitored system regardless of the application program used and to preclude the monitoring system from being removed or bypassed by unauthorised entities, embedding the requisite sensors at the kernel level is highly desirable.

### *3.1 Microsoft Windows file system structure*

The Microsoft Windows NT family of operating systems (including Windows 2000, XP, and Vista — the following uses the term Windows NT as a generic term unless a feature or behaviour specific to a given version is noted) [2, 3] exposes several APIs via environmental sub-systems. While these APIs are largely procedural in nature, the internal processing is asynchronous and packet-based in nature. Regardless of which environmental sub-system is used, the I/O operation eventually results in a call to the system service dispatcher in kernel mode. This dispatcher handles the distribution of the operations into the various kernel components. For the discussion of file system mechanisms, only some components are of interest.

Besides the I/O manager, the Windows management instrumentation (WMI), plug and play (PnP) manager, and the power manager components (these appeared beginning with Microsoft Windows 2000) are also relevant for device level operations. The central component for file system operations, however, is the I/O manager. It creates I/O request packets (IRPs) from incoming requests (with the exception of Fast I/O, see below) and ensures that all drivers for which an IRP is relevant are called with the IRP in the proper sequence. Each IRP sent to a kernel-mode driver represents a pending I/O request to that driver. An IRP will remain outstanding until the recipient of the IRP invokes the `IoCompleteRequest()` service routine for that particular IRP. Invoking `IoCompleteRequest()` on an IRP results in that I/O operation being marked as completed, and the I/O manager then triggers any post-completion processing that was awaiting completion of the I/O request. Each request must be completed exactly once.

This mechanism lends itself to a layered processing approach in which IRP messages are cascaded across several

driver layers (possibly with additional IRP messages created during the course of processing at lower levels). As a side effect of this architecture, one can alter the functionality of the operating system by interposing additional layers in the driver stack. One example of such an interposition is shown in Fig 1. The placement of the filtering layer in Fig 1 has the advantage of such a module being able to intercept and operate on generic (file-system-independent) operations from upper operating system layers; this type of filter is called a file system filter driver, and represents an approach also commonly found in antivirus and encryption software.

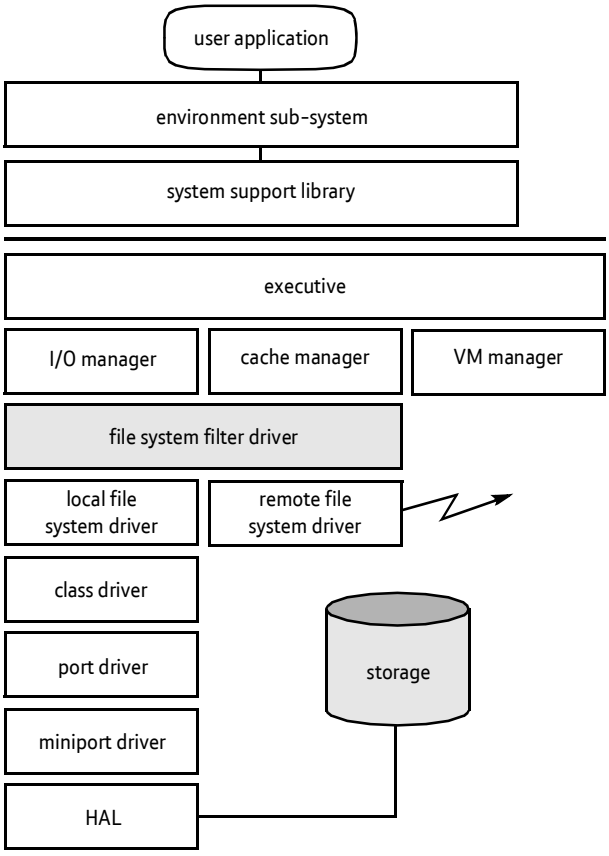


Fig 1 File system interception mechanism for the Microsoft Windows NT family of operating systems.

Microsoft Windows NT does not fully adhere to the packet-based I/O model for all types of driver, though. A special case exists in the case of file systems, and therefore also for file system filter drivers. This exception is the Fast I/O mechanism; here the I/O manager, cache manager, and the various file system implementations (if they support this mechanism) interact by means of explicit cross-module calls instead of creating IRP messages (see Fig 2). This performance enhancement adds considerable complexity to the design of any file system filter drivers, since additional communication paths must be handled. While it is possible for a driver to signal that Fast I/O is not supported, this results in an unacceptable performance degradation. The reason for this is that instead of the Fast I/O call, an equivalent call in the form of an IRP must be created by the

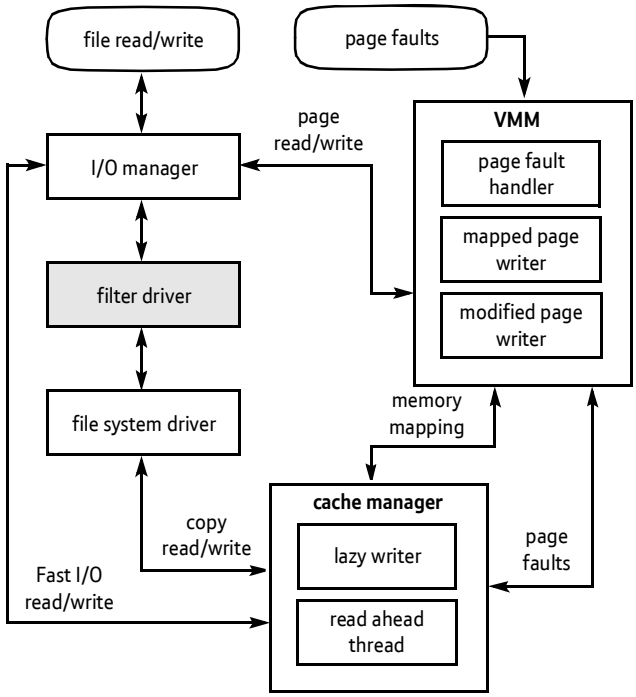


Fig 2 Interactions between file system components.

dispatcher. The creation of this IRP causes an additional performance penalty on top of the relative performance degradation incurred by not using Fast I/O. Moreover, once a filter driver has indicated this lack of support for Fast I/O, all subordinate lower-level drivers are also no longer confronted with Fast I/O.

Microsoft Windows NT uses the filter driver mechanism itself — not only for file systems, but also to support additional functionality that is optional or can be made available for different file system types with a single driver. One example which was provided by the core operating system beginning in Microsoft Windows 2000 is the single instance store (SIS) file system filter driver (that conserves disk space by removing multiple copies of a file and replacing them with links to a single shared copy in a common directory). Another application example of a file system filter driver is a malware scanner — again, this type of application requires access to file system semantics.

### *3.2 Instrumentation driver architecture*

The instrumentation driver does not require modifications to the file systems it is observing, which eliminates several potential problems such as maintaining cache and metadata storage consistency (see, for example, Wolthusen [4]). However it must be capable of monitoring all file systems that are mounted both statically and dynamically (e.g. in the case of a USB pen drive).

To ensure that this is the case, all file systems on a node must be intercepted and brought under the control of the instrumentation driver. This is achieved by registering a

callback function with the I/O manager which is called whenever a file system is loaded, which ensures that the filter driver can attach itself to all file systems, even those that are loaded dynamically at some point after booting. Dynamic loading of file systems can, for example, also occur when removable media (e.g. a UDF-formatted DVD disc) are loaded. An implication of this is that the `Filter` driver must be loaded prior to all file systems. This can be achieved by assigning it either to the `Filter` driver group or associating the necessary tag value with it in the registry settings for the driver load sequence. The only file system for which such a filter driver is not notified of a load event is the `Raw` file system (permitting access to the raw device without any file system semantics interpretation). However, given that only privileged processes such as database management systems require such access, monitoring can also be replaced by inhibiting user access to the file system type. Once the notification callback is called, the filter driver can attach itself to the file system or file system recogniser, respectively, and is then able to intercept the file system control requests (with the minor functions `LoadFS` and `MountVolume`) and attach itself to mounted volumes. Once it is attached to a mounted volume, the filter driver can intercept all necessary I/O requests.

The primary interception mechanisms for monitoring are the observation of relevant metadata and file read and write access, which are covered by the `IRP_MJ_READ` and `IRP_MJ_WRITE` IRPs.

For obtaining metadata information, the IRPs `IRP_MJ_QUERY_INFORMATION` and `IRP_MJ_SET_INFORMATION` along with ancillary IRPs such as the `IRP_MJ_DIRECTORY_CONTROL` and `IRP_MJ_QUERY_DIRECTORY` requests must be intercepted.

Both metadata acquisition and interception are, however, triggered by the processing of `IRP_MJ_CREATE` requests. This IRP is issued when a file is accessed for the first time (i.e.not just for file creation) by an upper level function. The instrumentation driver can subsequently issue a number of additional IRPs itself to gather all requisite information (e.g. `IRP_MJ_QUERY_INFORMATION`) for later processing. The metadata captured in this context includes the process (user) accessing the file, its canonical location within the file system, timestamps for access modes (reading and writing), and the type of access desired.

The time of first access is also used to determine whether the metadata (and subsequently also the actual file content) is to be recorded at all. This step is dictated by the way the metadata is accessed (and subsequently cached) within the operating system. This can be configured dynamically by the security administrator through the use of configuration files that are cached at the kernel level. The reason for this mechanism (compared to, for example, retaining data in the system's registry database) is primarily the performance impact of switching between kernel and user modes as well as limitations on such switches depending on the type and status of a given IRP and possible conflicts in accessing the registry with other (user) processes. Changes in configuration are therefore to be effected only through an explicit signal to the filter driver advising it to refresh the configuration at the earliest possible time. A second deselection criterion for metadata is caused by the fact that Windows NT uses the same IRP not only for files but also for a large variety of other objects, including transient entities such as named pipes and shared memory segments. Since these typically do not share file semantics and may be performance sensitive, they should be omitted from interception.

### *3.3 File shadowing mechanism*

Two types of data must be retained in the audit trail. The first consists of the metadata associated with the file access. This is accomplished by using the list of open file handles maintained by the operating system and creating a new record if a process first accesses a file to avoid an excessive number of records being generated. Depending on the threat model it can also be desirable to suppress certain spurious requests generated by the operating system (e.g. the Explorer file system viewer or the common file open dialogues) as these tend to enumerate metadata to generate listings. Similarly, misbehaviour of the Windows 2000/XP network share access mechanism can also trigger full `IRP_MJ_QUERY_INFORMATION` enumeration behaviour for all contained files when accessing directories via shares. While it is conceivable that suppression of such records can also mask misbehaviour, positive identification of the relevant processes (and disallowing users to, for example, install shell extensions for the Explorer program) can minimise this risk.

Metadata records are stored as a single page (4 kB) per access record, indexed by the canonical file name, subject identity, and access time; only one record is written for a given process indicating the first time a file is being accessed (the actual index is built later in a user space process, not described in this paper). The metadata record also contains a reference to the location of the copy of the individual file. This location (file name) is the full canonical name of the original file, relocated to the storage file system (i.e. with an additional prefix identifying the storage file system).

However, given that files may be changed arbitrarily between the recording of metadata and an analysis of an entity's behaviour, it is necessary to shadow the files accessed by the entities monitored. This can be effected by mirroring the files read and written in secondary storage. However, to avoid undue performance impact, a key objective of the storage component for the file system mechanism is the minimisation of required operations while executing I/O operations, resulting in a somewhat inefficient storage layout. Both metadata and individual files are

allocated as sparse files. This feature is available only beginning with Microsoft Windows 2000 in version 5.0 of NTFS and later; it also requires that the sensor storage is placed on an NTFS file system, which, however, is also highly desirable for security reasons alone. Using sparse files results in significant savings in storage required. The storage can be allocated on any file system, but to retain the invisibility of the system to the user, this is best achieved by allocating one or more separate file systems to it and hiding it (at the level of the file system filter driver) from the remainder of the operating system, and hence from both observation and manipulation by users and application programs. For reasons of performance (since the mechanism makes use of the VM architecture), it is desirable to locate these shadowing volumes on local storage (see Wolthusen [1] for a discussion of performance benchmarks).

All file systems of interest, including dynamically loaded file systems, must be intercepted; to this end, a notification callback for the file system minor functions `LoadFS` and `MountVolume` are used. The only exception to this mechanism, the `Raw` file system, is generally blocked for regular user processes since it permits circumvention of operating system access control mechanisms. Once the interception mechanism has determined that a file is accessed by a process of interest, metadata is stored in a database file with fixed record size indexed by the canonical file name (which must also include file system specific structures such as alternate data streams supported by NTFS), subject identity, and last access (indices are retained in main memory).

Subsequent read access for these files is then augmented by a write process that maps each page read occurring for a write operation into a write page request for the storage sub-system. This is achieved by performing a memory mapping between the respective uses, resulting in efficient caching and a reduced number of actual disk operations incurred (i.e. at most one page write operation occurs for an arbitrary number of page read operations provided that no modification of the pages read occurs on the part of the reading process). The result is a snapshot of all pages read by the process. However, since the analytical processes require not only the actual pages read but also context to re-establish the semantics of file contents, backfiling of pages unread by the user process is required.

While maintaining a separate cache of pages read by a process would be feasible (mirroring existing data structures), it is not reliably possible (particularly for remote file systems and removable media) to detect future operations on files, so shadowing entire files is required even if only partial read operations have been effected. This is accomplished by creating a worker thread that replicates the unread pages by initiating page reads itself, independent of the user process behaviour. The computational complexity, and, in particular the I/O load of this operation, can be reduced significantly by utilising the fact that the cache manager will perform a predictive read-ahead itself. By reusing the pages already cached and inducing predictive read-ahead, both total I/O operations and cache memory allocation can be minimised — however, there is a significant cost incurred in the additional page writes that must be scheduled.

## 4. Network stack instrumentation

In addition to traditional file-system-based operations, information retrieval from network-based databases and services are also of particular interest. Using additional network-based sensors, this additional source of information can also be subjected to surveillance and subsequent analysis. Although there exists a large number of network as well as application protocols for such retrieval systems, the Internet protocol and HTTP (and HTTPS) protocol clearly dominate as network and application protocol, respectively.

The mechanisms for interposition of an interception mechanism (with additional transparent in-line proxying for HTTPS (TLS) connections) have been described elsewhere [5, 6]; by inserting kernel modules and driver components at several locations within the Windows NT network protocol stack, all inbound and outbound network traffic can be observed transparently without affecting application programs — for a discussion of the specific adaptations required for extracting sensor data refer to Wolthusen [1].

## 5. Extraction of textual data

Sensor data extraction can occur in multiple steps on demand from one of the detection mechanisms (which may cache the extracted information separately) — all of which are performed by a system service operating in the background that does not communicate with regular user processes. However, this process can still trigger file backfilling in the file system filter driver through IOCTL calls (in case the caching mechanism has not yet backfilled a file selected for data extraction). The first extraction step is the identification of possible outer encodings (e.g. file compression, MIME transport encodings), followed by the determination of file type — this must be performed heuristically by analysing the start of the file for either explicit file type information or sufficient data to deduce file type, additional information may be gained from file names or auxiliary data streams. Based on the file type classification, files can be forwarded through an extensible dispatcher system to media-specific extractors (e.g. for textual, image or audio data), although the following section is concerned exclusively with textual data.

For the class of textual data representation formats — as for all other media types — there exists a large number of file formats and encodings of which pragmatically only a limited selection can be addressed. Other than for plain text

(for which the encoding may still need to be determined if data is not presented in ISO 646 or 10646 format or the chosen file format makes the encoding explicit), a translation filter is still required. These filters can (partially) parse mark-up languages such as SGML, XML, and HTML, although in the latter case extraction is limited to removing mark-up language since attributes cannot be extracted reliably as is the case for SGML and XML. Of particular interest for XML document decoding is the open office document type descriptor (the open document format is currently undergoing standardisation within the OASIS group). To avoid custom development of complex filters that need to be adapted frequently to version updates, proprietary text formats (such as those used by earlier Microsoft Office formats) can be reliably converted into XML and from there into plain text using the OpenOffice import filters. This does, however, incur a considerable performance cost compared to a proprietary extraction mechanism, but with significantly better results than in the case of simple extraction of printable text strings. The mechanism as presently defined is, however, limited to largely unstructured text since it eliminates the schema information in case of structured mark-up. Similar issues also exist with the popular Adobe PDF format; however, here elements of the GNU XPdf project can be used to extract plain text and encodings from within the extractor service. Regardless of the preceding steps, output of the text data extractor is a normalised ISO 10646 plain text data stream that does not contain metadata. Other formats nominally containing textual data, particularly Adobe PostScript data, may not be readily suited for conversion in all cases since certain PostScript output filters do not readily retain the context for words or even individual characters, making the extraction mechanism severely error-prone.

## 6. Analysis

The analysis to be performed over the documents and meta-data gathered by the mechanisms described above is limited in scope to identifying terms and conceptual units that may lie outside the regular remit of an entity, typically acting as a pre-filtering and alerting system for human expert intervention. For this role, natural language understanding is not required as such; rather, it is sufficient to apply statistical techniques for anomaly detection commonly found in signal processing and intrusion detection to data sets preprocessed by natural language processing algorithms. This implies the distinction of several types of behaviour:

- document retrieval and processing within a given tasking can be characterised by a restriction to finite set of concepts (clusters),
- conceptual drift results in the inclusion of a limited number of related concepts into the tasking concept set over time — such behaviour must be monitored to avoid a knowledgeable individual inducing slow, deliberate drift, thereby thwarting anomaly detection,
- abrupt changes in the constitution of the concept set with limited overlap, that stabilise once the shift has occurred, can be assumed to indicate a new tasking — this may also be confirmed by supplemental information.

As noted above, detection of anomalies requires that pertinent concepts can be identified automatically; this must occur at several levels. A prerequisite step is the creation of a concept dictionary along with thesauri for synonyms and related terms. While an initial dictionary, particularly of task-related terms, must be built up manually, a number of techniques exist that permit automatic extension and derivation of such databases [7—13]; depending on the types of documents to be processed, this may also require extension to multilingual corpora [14, 15]. Based on such concept groupings, a second problem that needs to be automated as far as possible is the separation of conceptual clusters through text categorisation [16, 17]. A number of approaches have been proposed for this technique, including linear classifiers [18], context-sensitive learning mechanisms [19], Bayesian techniques [20], and decision trees [21], although typically a combination of techniques and algorithms is used, frequently based on boosting-based classifier committees, support-vector machines and regression methods. Such classifiers exhibit adequate performance even for very large category sets [22]. In particular, boosted Bayesian networks have been successfully used on large corpora of documents (approximately $10^5$) and categories (approximately $10^4$) over extended periods [23—26]; by combining multiple properties such as term properties, relations over terms and documents, and document properties (e.g. location in the file system, metadata attributes) and boosting multiple weak hypotheses, separation can be obtained with limited term occurrences [17].

Although the resulting data set is still of considerable dimensionality, a reduction of several orders of magnitude can be achieved by categorisation as discussed above. The resulting data set can now be subdivided manually (e.g. using techniques from formal concept analysis [27, 28]) to achieve further dimensionality reduction; alternatively, analytical techniques suitable for such high-dimensionality systems can be employed. In any case, one additional dimension (time) must be added to the data set.

Given the above processes, the behavioural anomalies described earlier can now be identified using statistical analysis techniques also commonly used in intrusion detection. A technique particularly suited for such analysis is multidimensional scaling (MDS) [29—31] followed by identifying centres of gravity for identified clusters and outliers from these clusters [32—34]; this can occur either automatically (based on fixed scaled thresholds) or in preparation for visual inspection (although in this case the mapping of a high dimensional space on to a two- or three-

dimensional plot does not necessarily preserve structural properties such as linear separability of categories).

For each two documents of the observation set (obtained by restricting the data set to a given entity (individual) and a duration) $i$, $j$, a proximity metric $P_{ij}$ is defined such that $P_{ij}$ is smaller if the similarity between $i$ and $j$ is larger. A configuration $X$ is constituted by $n$ points in an $m$-dimensional space and can be considered an $n \times n$ matrix of the co-ordinates of the $n$ points along $m$ axes of a Cartesian co-ordinate system. The distance of points $i$ and $j$ in $X$, $d_{ij}$, can now be computed as:

$$d_{ij} = \left( \sum_{a=1}^{m} \left| x_{ia} - x_{ja} \right|^{m} \right)^{\frac{1}{m}}$$

where $x_{kl}$ is the co-ordinate of point $k$ along the axis $l$ of the co-ordinate system. In the simplest (metric) case, the identity mapping is used to map the proximity measure ($f(p_{ij}) = p_{ij} = d_{ij}$) by way of a Minkowski distance; this, however, is justified only if the dissimilarity measure can be embedded in a metric space $(K, \delta)$ where $K$ is a set of points with $x, y, z \in K$ and $\delta(x,y)$ is a function $\delta: K \times K \rightarrow \mathbb{N}_0$ such that:

$$\delta(x, y) = 0 \Leftrightarrow x = y \qquad \textit{minimality} \qquad \ldots (1)$$

$$\delta(x, y) = \delta(y, x) \qquad \textit{symmetry} \qquad \ldots (2)$$

$$\delta(x, z) \leq \delta(y, x) + \delta(y, z) \qquad \textit{triangle inequality} \qquad \ldots (3)$$

Non-metric MDS [30, 35] employs arbitrary functions $f$ and merely assumes a monotonic relation between orderings of similarities and rank order of metric distances in a metric space; for the purposes of this discussion, however, metric MDS suffices. It should be noted, however, that these are merely basic examples of techniques that can be applied to the problem of identifying clusters and anomalies within the reduced semantic data space and may not necessarily provide optimum results.

## 7. Conclusions

The paper has described an efficient mechanism for shadowing the activity of users with regard to the handling of documents (focused on textual documents, but extensible to other media types) that may be considered abnormal but cannot be prevented entirely using access control mechanisms. The basic mechanism has many uses in an operational risk management system, from generating audit logs for compliance monitoring, forensic investigations and other purposes, to providing input to behavioural analysis functions. The latter application has been the main focus of the paper.

The extraction mechanism is embedded in the operating system and makes use of caching strategies for file system access by the operating system to provide a shadowing of both files (documents) and metadata that does not impede regular operations. Extraction of text and the application of clustering for concepts using natural language processing allows the subsequent application of multidimensional scaling and related processing techniques for anomaly detection which can occur either automatically or as a targeted preprocessing step for human intervention.

Future work will require a detailed usage pattern and performance analysis in operation, using various applications, since the file system access patterns (particularly of custom application) can have significant adverse impact on overall performance that is predicted on average case behaviour. While the textual and conceptual processing steps and analysis were not the focus of the work reported here, this area is clearly a subject for further research, particularly investigating the impact of using different and multilingual corpora on the accuracy of clustering and anomaly detection.

## Acknowledgments

Parts of the research reported here were conducted while the author was at the Norwegian Information Security Laboratory of Gjøvik University College, Norway and Fraunhofer-IGD, Darmstadt, Germany.

## References

1 Wolthusen S: 'Molehunt: Near-line Semantic Activity Tracing', in Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop, United States Military Academy, West Point, NY, IEEE Press, pp 410—418 (June 2005).

2 Russinovich M E and Solomon D A: 'Microsoft Windows Internals', 4th Edition, Microsoft Press, Redmond, WA (2004).

3 Solomon D A and Russinovich M E: 'Inside Microsoft Windows 2000', 3rd Edition, Microsoft Press, Redmond, WA (2000).

4 Wolthusen S: 'Security Policy Enforcement at the File System Level in the Windows NT Operating System Family', in Proceedings 17th Annual Computer Security Applications Conference, ACSAC'01, New Orleans, LA, IEEE Press, pp 55—63 (December 2001).

5 Rademer E and Wolthusen S: 'Transparent Access To Encrypted Data Using Operating System Network Stack Extensions', in Steinmetz R, Dittman J and Steinebach M (Eds): 'Communications and Multimedia Security Issues of the New Century', Proceedings of the IFIP TC6/TC11 Fifth Joint Working Conference on Communications and Multimedia Security, CMS'01, Darmstadt, IFIP, Kluwer Academic Publishers, pp 213—226 (May 2001).

6 Wolthusen S: 'Tempering Network Stacks', in Proceedings of the NATO RTO Symposium on Adaptive Defense in Unclassified Networks, Toulouse, France, NATO Research and Technology Organization (April 2004) .

7 Crouch C J: 'An Approach to the Automatic Construction of Global Thesauri', Information Processing and Management, 26, No 5, pp 629—640 (1990).

8 Berry M W (Ed): 'Survey of Text Mining: Clustering, Classification and Retrieval', Springer-Verlag, Heidelberg (2003).

9 Senellart P P and Blondel V D: 'Automatic Discovery of Similar Words', in Berry M W (Ed): 'Survey of Text Mining: Clustering, Classification and Retrieval', Springer-Verlag, Heidelberg (2003).

10 Fellbaum C J (Ed): 'WordNet: An Electronic Lexical Database', MIT Press, Cambridge, MA (1998).

11 Lesk M: 'Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone', in Proceedings of the 5th Annual International Conference on Systems Documentation, Toronto, Ontario, Springer-Verlag, pp 24—26 (June 1986).

12 Banerjee S and Pedersen T: 'An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet', in Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics, CICLING-02, Mexico City, Mexico, Lecture Notes in Computer Science, Vol 2276, Springer-Verlag, pp 805—810 (February 2002).

13 Banerjee S and Pedersen T: 'Extended Gloss Overlaps as a Measure of Semantic Relatedness', in Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence IJCAI-2003, Acapulco, Mexico, Morgan Kaufmann, (August 2003).

14 Pazienza M T (Ed): 'Information Extraction: Towards Scalable, Adaptable Systems', LNAI 1714, Springer Verlag, Heidelberg (1999).

15 Somers H: 'Knowledge Extraction from Bilingual Corpora', in Pazienza M T (Ed): Information Extraction: Towards Scalable, Adaptable Systems', Lecture Notes in Artificial Intelligence, Vol 1714, Springer Verlag, Heidelberg (1999).

16 Manning C D and Schütze H: 'Foundations of Statistical Natural Language Processing', MIT Press, Cambridge, MA (1999).

17 Sebastiani F: 'Machine Learning in Automated Text Categorization', ACM Computing Surveys, 54, No 1, pp 1—47 (2002).

18 Lewis D D, Schapire R E, Callan J P and Papka R: 'Training Algorithms for Linear Text Classifiers', in Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, ACM Press, pp 298—306 (August 1996).

19 Cohen W W and Singer Y: 'Context-Sensitive Learning Methods for Text Categorization', ACM Transactions on Information Systems, 17, No 2, pp 141—173 (1999).

20 Makoto Iwayama T T: 'Hierarchical Bayesian Clustering for Automatic Text Classification', in Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI-1995, Montreal, QC, Morgan Kaufmann, pp 1322—1327 (August 1995).

21 Apté C, Damerau F and Weiss S M: 'Automated Learning of Decision Rules for Text Categorization', ACM Transactions on Information Systems, 12, No 3, pp 233—251 (1994).

22 Yang Y, Zhang J and Kisiel B: 'Text Categorization: A Scalability Analysis of Classifiers in Text Categorization', in Proceedings of the 26th Annual International ACM SIGIR conference on Research and Development in Information Retrieval, Toronto, ON, ACM Press, pp 96—103 (July 2003).

23 Fuhr N: 'A Probabilistic Model of Dictionary Based Automatic Indexing', in Proceedings of RIAO-85, First International Conference 'Recherche d'Information Assistee par Ordinateur', Grenoble, France, pp 207—216 (March 1985).

24 Fuhr N, Hartmann S, Lustig G, Schwantner M, Tzeras K and Knorz G: 'AIR/X: a Rule-Based Multistage Indexing System for Large Subject Fields', in Proceedings of RIAO-91, Third International Conference 'Recherche d'Information Assistee par Ordinateur', Barcelona, Spain, pp 606—623 (April 1991).

25 Knorz G: 'A Decision Theory Approach to Optimal Automatic Indexing', in Proceedings of the 5th Annual ACM Conference on Research and Development in Information Retrieval, Berlin, pp 174—193, ACM Press (May 1982).

26 Tzeras K and Hartmann S: 'Automatic Indexing Based on Bayesian Inference Networks', in Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Pittsburgh, PA, ACM Press, pp 22—35 (June 1993).

27 Ganter B and Wille R: 'Formal Concept Analysis — Mathematical Foundations', Springer Verlag, Heidelberg, Germany, originally released in German as 'Formale Begriffsanalyse — Mathematische Grundlagen' (1998).

28 Priss U: 'Linguistic Applications of Formal Concept Analysis', in Proceedings of the First International Conference on Formal Concept Analysis, ICFCA 2003, Lecture Notes in Artificial Intelligence, Vol 2276, Springer-Verlag (March 2003).

29 Kotz S, Johnson N L and Read C B (Eds): 'Encyclopedia of Statistical Sciences, 5, John Wiley & Sons, Inc, New York (1985).

30 Kruskal J and Wish M: 'Multidimensional Scaling', Vol 07-011 of Safe University Paper Series on Qualitative Applications in the Social Sciences, Sage Publications, London, UK (1978).

31 Young F W: 'Multidimensional Scaling', in Kotz S et al (Eds): 'Encyclopedia of Statistical Sciences', Vol 5, John Wiley & Sons Inc, New York (1985).

32 Kruskal J B: 'The Relationship between Multidimensional Scaling and Clustering', in van Ryzin (Ed): 'Classification and Clustering', Academic Press, New York (1977).

33 Strehl A: 'Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining', PhD thesis, University of Texas at Austin, Austin, TX (2002).

34 van Ryzin J (Ed.): 'Classification and Clustering', Academic Press, New York (1977).

35 Kruskal J B: 'Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis', Psychometrika, 29, pp 1—27 (1964).


Stephen Wolthusen joined the information security group at Royal Holloway as a lecturer and also holds an associate professorshp at the Norwegian Information Security Laboratory at Gjøvik University College, Norway. Before joining the faculty of Royal Holloway he worked at the Fraunhofer-IGD laboratory in Darmstadt, Germany as a deputy division chief to which he is still affiliated as a senior scientist. He is author of several books, has edited multiple conference proceedings volumes and also holds several German and international patents. His primary research interests are in the areas of information assurance and the use of formal methods for modelling, specification, and verification as well as models and analytical techniques for the protection of critical infrastructures. He received both his Dipl-Inform degree in computer science and a PhD in theoretical computer science from TU Darmstadt, Germany.