Sensitivity Labels and Invisible Identification Markings in Human-Readable Output

Christoph Busch[a] and Stephen D. Wolthusen[a]

[a]Fraunhofer-IGD, Security Technology Department, Darmstadt, Germany

ABSTRACT

This paper presents a mechanism for embedding both immediately readable and steganographically hidden information in human-readable output, particularly in hard copy format. The mechanism is embedded within a domain inaccessible to unprivileged users in the operating system's Trusted Computing Base. A realization is presented which permits the embedding of such markings in arbitrary printing systems under the Microsoft Windows NT family of operating systems.

Keywords: Labels, Digital Watermarking, Hard Copy

1. INTRODUCTION

Labeling human-readable output with sensitivity labels is a requirement for implementations conforming to classes in the TCSEC B division[1] and its successors (e.g. the Common Criteria LSPP[2,3]). Since the creation of the requirements, however, developments particularly in the area of printing hard copies have made it difficult to embed such labels since line-oriented printing devices have been supplanted by page-oriented devices with a variety of page-description languages which do not lend themselves as easily to the required postprocessing steps.

The need for adequate labeling is also prevalent in civilian application areas; while it is generally not feasible to restrict or prohibit the creation of hard copy as that would be considered an undue burden on the work flow, the risks associated with uncontrolled and careless handling of sensitive printed material are considerable. This risk not only includes leaving sensitive material unguarded in insecure locations but rather extends beyond actual use in that such material is discarded intact without adequate prior shredding, burning, or similar destruction techniques.

In other areas, such as multimedia data – e.g. audio and video data – the issue of applying sensitivity labels must also be addressed even though the relevant standards do not as yet address these issues.

Another issue that needs to be addressed is the automatic and reliable creation of auditable records for the creation of hard copies of sensitive material. Printing accounting systems currently in use can only provide records of the creation of hard copies and at best provide an application and document name in such records. The latter are not mandatory and are easily circumvented or fabricated.

In addition to sensitivity labels, it is generally desirable — possibly even for public materials — to have an automatable capability for identifying a document unambiguously even if only a fragment of a single page of such a document is found. While it is possible to add identification markings to individual pages, conventional mechanisms such as bar codes are highly localized and cannot be used e.g. in case only a figure or a photograph of a labeled document is reproduced elsewhere and needs to be identified. By providing a mechanism to embed not only the identity of the document but also that of the user associated with the process initiating the creation of the hard copy, a location-independent audit record is provided which can also be back-linked with other audit events to provide insight into the distribution and handling of sensitive material.

Further author information:
E-mail: {busch|wolt}@igd.fhg.de, Telephone: +49 6151 155 {147|539}, Address: Rundeturmstr. 6, Darmstadt 64283, Germany

The characteristics outlined above can be implemented by means of a filtering mechanism that is interposed into the operating system in such a way that any and all print output must pass through the filtering mechanism. This is desirable for several reasons; the first one being that interception at the privileged operating system level removes the possibility for tampering with the labeling facility by unprivileged users (assuming that the separation between privileged and unprivileged execution is effectively enforced by the remainder of the operating system). Another important reason is that embedding at the operating system level while retaining the external application programming interfaces implies that any application program using these interfaces will run unmodified and unaware of any additional processing.

The second component required for fulfilling the requirements as discussed above is digital watermarking. Documents typically consist of a number of different media types such as text, charts or similar vector-based material, and raster images where the composition will vary from page to page of the printed output. Each of these media types requires specific handling to ensure that the requirements for robustness and visibility are met; the naïve approach of converting the output to a raster format and applying a raster marking technique is both inefficient and produces visible artifacts. Since the requirement is that the technique must be able to recover the label even from a fragment of a document, this implies that all of the media types present on a printed document must be handled.

Section 2 gives the requirements for such watermarking algorithms while section 3 describes the requirements for the environment in which such watermarking (with the addition of visible marking as applications warrant) must take place; sections 4 and 5 discuss characteristics for the content identification markings. Section 6 provides an implementation outline.

2. WATERMARK REQUIREMENTS FOR A LABELING SYSTEM

The following requirements must be met by digital watermarking algorithms useful for labeling with sensitivity and identification as primary application areas and hard copy as the principal output medium:

1. The watermarks must be robust against D/A — A/D processing.

   This implies that the algorithm must embed the marking in the carrier signal; it also introduces a robustness requirement against jitter.

2. The watermarks must be robust against cropping.

   A fragment of the original human-readable output should suffice to recover all or part of the marking.

3. The watermarks must be robust against affine transforms.

   Some processing steps applied to a hard copy (e.g. automatic paper format resizing applied by some printers) can introduce global affine transforms. This is opposed to local affine transforms such as those produced by irregularities and imperfections in photocopying which are subsumed under requirement (1).

4. Payload sizes must be ≥ 64 bit.

   The payload must be sufficiently large to carry information such as the classification and categories, identification of origin, timestamps, the identity of the subject initiating the human-readable output, and the owner of the document. While all this information can be stored in database facilities, it is necessary to have a sufficiently large domain from which entries can be selected. Depending on the need for blind detection, more than one payload entry may be necessary.

5. Blind detection schemes must be used.

   Since the original signal typically does not exist in this application scenario, the watermark must be recoverable without taking recourse to a comparison with a reference signal.

6. Visibility of the watermark should be low to imperceptible.

   This requirement is due to ensure that the signal degradation introduced by the marking does not adversely affect the usability of the human-readable output.

These requirements are kept qualitative since the precise metrics for which level of attack a marking must survive (e.g. whether 50% or 90% of the original may be cropped in case of requirement 2) depend on the application area.

In case of print output, there are several types of objects that can be identified for marking:

- Raster images
- Vector graphics
- Individual lines of text
- Text paragraphs

However, vector-based are typically too sensitive for marking except under well-defined circumstances[4]; in addition, blind detection of such markings in the face of modifications such as cropping and affine transforms is difficult at best without semantic knowledge.

Watermarking techniques for raster images are well established[5]; depending on the algorithm used a certain minimum size for raster images must be used. For individual lines of text, several algorithms also exist,[6] the same is true for text paragraphs.[7]

The marking techniques themselves are not discussed here as — notwithstanding differences in performance and robustness among them — any technique fulfilling the requirements listed above including the ones cited are sufficient for the purposes discussed here.

3. ENVIRONMENTAL REQUIREMENTS FOR A LABELING SYSTEM

The requirements for the labeling and marking mechanism are straightforward:

1. The mechanism must not be bypassable by unprivileged users
2. The mechanism must not be bypassable by applications
3. The mechanism must not affect the interface presented by the operating system to application programs.

Requirements 1 and 2 are obvious. Fulfilling these requirements can typicall be accomplished by using the resource access control mechanisms of the operating system to restrict access to interfaces other than the printing subsystem; only the subsystem with the embedded labeling mechanism may access these interfaces. Requirement 3 is called for since it is not economically feasible to modify COTS application programs to interoperate. If the marking system is part of security functionality to be evaluated, the entirety of the filtering and actual labeling and marking mechanisms must be part of the Target of Evaluation Security Functions.[2]

4. IDENTIFICATION MARKINGS

As noted in section 1 there are several purposes for watermarking hard copies. The embedding of initiator identification requires that the subject initiating printing can be identified; this will generally be a process abstraction of the operating system which itself can be connected to a human user (alternatively, a system process may initiate such a process and then represents the highest abstraction level entity which may be associated with the printout). The payload for the marking process is then the highest abstraction level subject that can be found.

If possible (such as is e.g. the case with ([5])) the watermarks should be embedded using a secret key; the use of multiple secret keys (i.e. ideally one per node which performs the embedding) is desirable to limit the exposure of compromised key material. The obvious drawback in such a case is that the number of hypothesis tests becomes linear in the number of nodes which are suspected of being the source of an embedding. As described

here, all marking techniques use secret keys for embedding and to ensure minimum exposure a per-node key scheme is used.

This mechanism can be applied for isolated systems; as most non-trivial applications require networked systems it is highly desirable to identify subjects (e.g. processes, applications, users) consistently across individual nodes in such a network even if the systems do not share the same basic operating system. This is accomplished by additional operating system extensions not discussed here[8*] which associate each subject with a subject type vector and a subject identity vector. These vectors are abstractions which are modeled by properties of host operating systems and for which a bijective mapping between the formal logic modeling primitives and the host operating system constructs exists in each case. Since payload size is limited and the vectors may be of arbitrary length; a compact representation must be used.

Such a compact representation is achieved by applying a cryptographic hash function to the vectors and embedding the hash value in lieu of the full subject type and identity vectors. From the properties of the hash function[10] one can easily see that selecting any arbitrary but fixed sequence of bits from the output bits generated by such a function the individual bits selected retain the properties of the entire function, particularly the avalanche effect.

Assuming that m is the number of unique vectors, an ideal cryptographic hash function, and a payload size of k bits, the number of compact representations is $N = 2^k$; the ratio of the number of compact representations without collisions to the total number of vectors is $\prod_{i=0}^{m-1}\left(1-\frac{i}{N}\right)$. Since $1 - x \leq e^{-x}$ (for $x > 0$) we have

$$\prod_{i=0}^{m-1}\left(1-\frac{i}{N}\right) \leq \prod_{i=0}^{m-1} e^{-\frac{i}{N}} = e^{-\sum_{i=0}^{m-1}\frac{i}{N}} = e^{-\frac{m(m-1)}{2N}}$$

Thus, the probability of a collision after inserting $\sqrt{2N}$ keys is $1 - \frac{1}{e}$. For a 32 bit payload this implies a collision once for every 90,000 subjects; for a 64 bit payload one collision in 6 billion subjects can be expected, which we consider acceptable – hence the requirement for a 64 bit payload. Obviously, these calculations do not take into account that the payload may not be extracted faithfully; if this is to be avoided, the payload size must be increased further to accomodate error detection or correcting codes, possibly causing visible degradation of the output.

For (optional) timestamps the requirements for payload are considerably smaller; we consider one minute resolution entirely adequate; 26 bits are sufficient to encode an entire century at this resolution.

5. SENSITIVITY LABELS

Since only page-oriented printing systems need to be considered here, there is considerable freedom in the placement, type, and size of the visible sensitivity labels.

By default, this means that a textual representation of the overall sensitivity of the output based either on the properties of identifiable objects present on the page to be printed or on the properties of all objects accessed by the subject (process) prior to submitting the printing request will be placed on the logical top and bottom of the output depending on the assumed orientation of the page. Alternatively, the overall sensitivity for all individual pages in a job can be derived from the properties of all objects accessed by the subject (process) prior to submitting the print job.

6. IMPLEMENTATION

The implementation of the labeling system consists of three components. First, a mechanism is required that intercepts all data destined for printing, modifies this data, and subsequently continues processing in the printing subsystem with the modified data. This filtering mechanism is heavily dependent on the host operating system it is to be integrated in. In cases where several paths exist that can result in a printout, any path that is not filtered by this mechanism must be disabled. The filtering component must also create detailed audit logs of

*An earlier revision for such a mechanism was described in ([9])

operations involving the printing system, namely the time and date of a printout and the identity of the subject (i.e. in most cases the user) submitting the print job; additional audit information such as the objects the subject accessed prior to submitting the print jobs depend on the presence of additional security components. In most cases this functionality can be added to the host operating system without requiring source access to the operating system.

In case of the Microsoft Windows NT family of operating systems (referred to as Windows in the following unless noting features particular to a version), the printing architecture is a layered one. An overview of the Windows printing architecture is shown in figure 1. Here, user mode components are not permitted direct access to hardware[†] and must use the GDI32 (Graphics Device Interface) library to issue commands for displaying or printing data. The GDI32 operates in the user mode domain and communicates the requests to its counterpart in the kernel mode domain, the GDI engine.

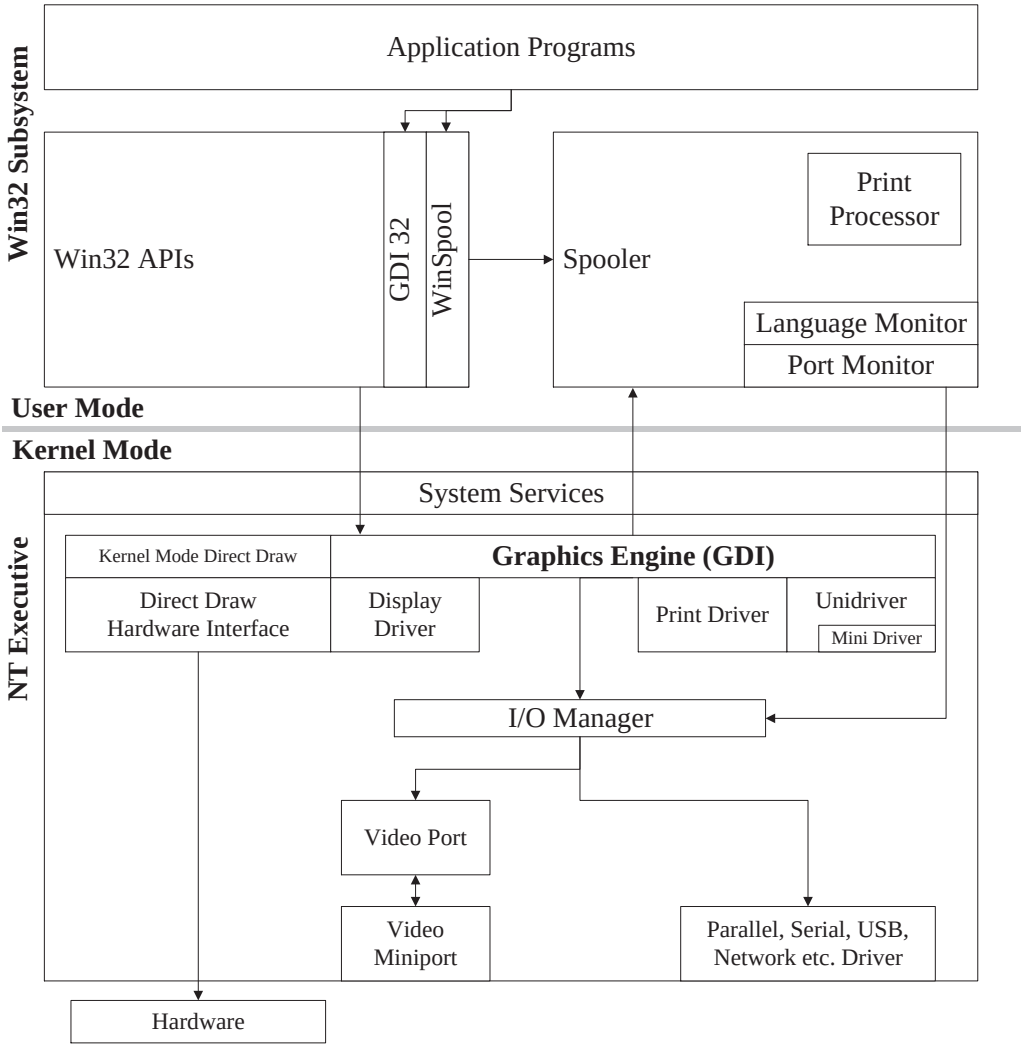

Figure 1: Windows NT/2000/XP Printing Architecture

The GDI engine is itself separated into several components. For translating the GDI commands into printer-specific commands, special printer drivers are required. The key feature here is that the printer drivers do not

[†]with the exception of Direct Draw for video output facilities; these have to be disabled for secure configurations

communicate directly with the hardware, rather they are only attached to the GDI Engine as shown in figure 1 and return the commands generated in reply to the generic DDI (Display Device Interface) requests to the GDI Engine which in turn sends the data to the I/O Manager for actual issuing of the printer commands.

There are several generic (class) drivers (e.g. for raster and Adobe PostScript capable printers) supplied by the operating system, support for specific devices in this framework requires only a mini printer driver which is instantiated by the class driver and communicates only with this class driver. In addition, device manufacturers can also supply self-contained drivers that do not rely on class drivers. All printer drivers are implemented as dynamically linked libraries (DLL).

6.1. Filtering Mechanism

To ensure that both system-supplied and custom drivers are properly intercepted, the filtering mechanism consists of a printer "wrapper" driver DLL (PWD) which supplies the same entry points as a normal printer driver and replaces the original DLL with itself while renaming the original DLL (see figure 2). The original DLLs are moved to a location inaccessible to the GDI engine under a temporary unique name.

The GDI engine will instantiate and call the wrapper DLL instead of the original driver since it is indistinguishable in both name and the entry points supplied once a request is detected. The wrapper can now either directly forward calls not relevant to the security functionality to the original library, invoking the same entry points it has itself been called with along with the parameters obtained from the caller, or it can intercept the calls and subject these to further processing whose results are eventually also forwarded to the renamed original driver.

This processing is to be performed in the Labeling and Marking Subsystem (LMS), the second major component in this scenario. At the start of a print job, the access control subsystem can be invoked which can in turn verify that all prerequisite rules for printing in this instance have been fulfilled by submitting the request for decision to the Externally Controlled Reference Monitor.[11] For the remainder of the print job, the LMS is invoked by the PWD for each page.

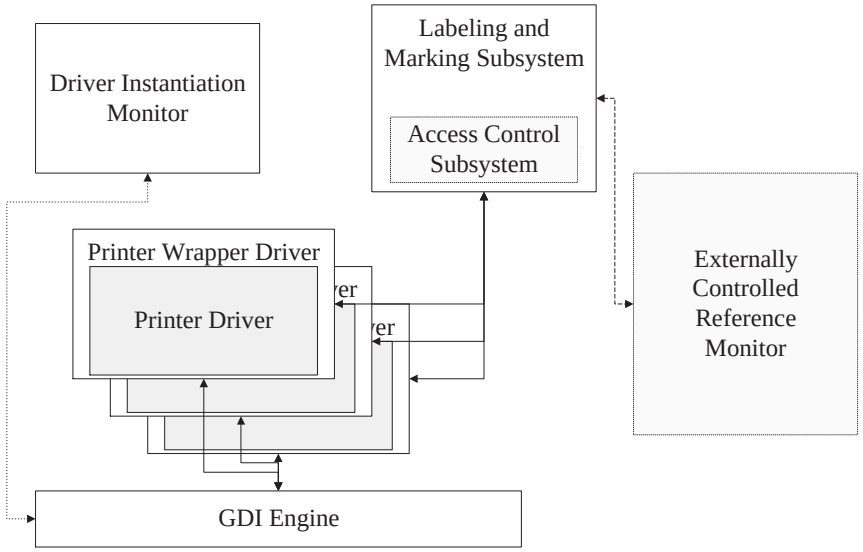

Figure 2: Labeling and Marking Components

Processing in the case of the Windows printing system consists of modifying the sequence of DDI calls for each page. The PWD must buffer the calls for each page in a meta-format and submit the DDI stream to the LMS (doing so for each call would also be possible, but would incur a sizeable performance penalty due to the doubling of call overhead).

The restriction to pages as the unit for marking is derived from the semantics of print processing under Windows; final processing for labeling and marking is possible only once a page has been finished by the print client since up to that point modifications to the entire page are still possible. As a result, some latency is introduced into the print process.

6.2. Driver Instantiation Monitor

The printers available to a system can change during runtime (e.g. if a USB printer is attached for which a driver exists in the system's driver database). It is therefore not possible to establish the PWDs at boot time or even during the installation of the labeling mechanism; restricting printing to disallow such plug-and-play operations would — while easier to supervise — impose an unnecessary burden on users.

Instead, the GDI subsystem must be monitored for events leading to the instantiation of a new printer driver. This can be accomplished by monitoring the Windows registry keys containing information on registered printers (`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print`) for attempted changes and redirecting the references on the fly. The reason for this is that the printing subsystem is not a proper device driver but rather derives from legacy implementations of prior operating system platforms. As a result, the standard course of action for such cases, namely the observation of instantiation of new `Device` executive objects with certain characteristics is not suitable. As modification of system module code is not desirable, observing the effects of new printer registration via the registry is the only viable option.

6.3. Labeling and Marking Mechanism

The LMS can now add human-readable sensitivity labels (or any other graphical output) to the page by simply adding DDI commands returned to the PWD for the given page. In case of the default markings, this is accomplished by overlaying a box with the background (paper) color at the logical top and bottom of each page in which the sensitivity label is placed. As this step occurs immediately before the finishing of each logical page, there is no possibility of the labels becoming obscured by the results of other DDI operations.

Digital watermarking is somewhat more involved than labeling since it requires that the input stream be modified instead of simple augmentation in the case of labeling.

However, raster images of usable size are treated as objects by the DDI and can thus be used directly as input for raster image watermarking. Since the marking does not affect the external features of the bitmaps relevant for placement of the objects in relation to other objects on a page (size, color resolution etc.), this permits direct replacement of original with marked bitmaps regardless of the marking algorithm used.

The same holds true for individual lines of text; DDI operates on lines at a time, although some application programs deliberately issue GDI calls for individual words. Depending on the size of the line received, the text watermarking module must therefore decide whether line-based algorithms are appropriate or whether word- or character-based algorithms must be used alone. To ensure that the overall layout is not perturbed, the external dimensions of such a marked block must not change. This requirement is lifted for paragraphs identified by the LMS as such and to which line-shifting watermarking techniques are applied to the extent that the external dimensions of the paragraph must not change.

One of the reasons for requiring robustness against affine transforms, including interpolation effects, is that print output (particularly raster images) will frequently be modified to accomodate different pixel aspect ratios.

To facilitate the integration of new and improved algorithms, the LMS is desigend to process the DDI commands in a modular fashion. The inbound DDI commands are categorized into three groups (vector, text, raster) and routed to the appropriate modules. Since the input and output in each case is a DDI command stream, the partial results can then be integrated again into a page representation. The main requirement for making this possible is that changes to the size and boundaries of the blocks processed are avoided.

6.4. Access Control and Auditing

When used in combination with an integrated layered protection mechanism ([8, 11, 12]), the filtering layer can also operate as an access control mechanism by denying print operations for which the subject or subjects are not authorized even if the host operating system itself does not support labeling.

The layered protection mechanism must provide information on the identity of the subject (i.e. an `EPROCESS` instance) which, as noted above, may be back-linked to a higher abstraction level subject. The subject is associated directly or indirectly through derivation with a security policy specifying the permitted behavior.

As a simple example based on the Bell-LaPadula (BLP) model,[13] a subject may be associated with a role consisting of security level and non-hierarchical categories; if a predicate for permission of a print job resolves this against the attributes of the individual objects (e.g. files, network connections) the subject entity has accessed during its lifetime (to continue the example, the list of hierarchical classification C_i and all non-hierarchical categories N_i^j for each object), printing is permitted. In terms of the BLP model, the subject must dominate all C_i, N_i^j or a specific predicate must permit printing under such circumstances.

Other items to be taken into account is the security level of the output device (e.g. network printers) or temporal restrictions specified by additional applicable predicates, all of which are be enforced by the externally controlled reference monitor system queried by the filtering layer and consolidated into the replies given to the respective subsystems upon querying for permission to perform an operation.

The process outlined above also provides detailed auditing information which can be collected and preprocessed locally; additional audit information is implicitly generated by each query to external reference monitors; the latter information can be dispersed across several instances and must therefore be consolidated prior to performing detailed analyses of the audit trail.

6.5. Applicability and Alternatives

While the mechanisms discussed here are functionally applicable under the consumer operating system family from Microsoft Corporation (e.g. Windows 95, 98), basic premises for the security of the system, namely the separation of domains between the marking system and components accessible to unprivileged users and the direct accessibility of output device interfaces to unprivileged users are violated and the labeling mechanism can be circumvented with very limited skills on the part of the adversary.

To ensure maximum effectiveness, the mechanism presented here must be embedded at all nodes to be protected; while some printer drivers support sending GDI/DDI meta-information to network printers, this is not a universal solution even if one is willing to accept that printers may be attached to the clients without central administration and control.

Modifying specific printer command languages is only moderately attractive; this either requires translation of the input stream into a common meta-format for labeling and marking with the complexity and performance impact inherent in such a mechanism or a redesign of the labeling and marking mechanism for each supported printer command language.

Due to a lack of a common printing architecture above the level of line-oriented devices the latter is precisely what is required in the case of Unix derivatives and other operating systems. Here the de facto standard for print systems is Adobe PostScript; however, the standard operating procedure is for applications to submit such data either to a print spooler or directly to an interface. However, given suitable configuration of interface permissions it is possible to force print output through a chain of print filters of the `lp` service. To ensure labeling and marking processing must occur only on one format, this may require double conversion, once each on input and output of the LMS system resulting in a considerable degradation of printing performance.

7. CONCLUSIONS

We presented a mechanism for embedding labels, particularly sensitivity labels, and digital watermarking into human-readable output. The mechanism is also applicable to other types of output such as audio and video output; the main issue in these two cases is the required processing speed to make real-time operation feasible.

By embedding the labeling and particularly marking into the operating system's domain one can ensure that the security policy's dictates are enforced regardless of user actions and without interfering with either user or application programs. We have also shown that the mechanism can be embedded into a COTS operating system, Microsoft Windows NT, without requiring access or modification to operating system source code and also demonstrated the effectiveness of the mechanism in conjunction with external reference monitors.

REFERENCES

1. United States Department of Defense National Computer Security Center, *Department of Defense Trusted Computer System Evaluation Criteria*, Dec. 1985. DoD 5200.28-STD.
2. ISO/IEC Standard 15408, *Common Criteria for Information Technology Security Evaluation*, version 2.1 ed., Dec. 1999.
3. National Security Agency Information Systems Security Organization, *Labeled Security Protection Profile*. 9800 Savage Road, Fort George G. Meade, MD, USA, Oct. 1999. Version 1.D.
4. C. Busch, E. Rademer, M. Schmucker, and S. Wolthusen, "Concepts for a watermarking technique for music scores," in *Proceedings of the Visual 2000/3rd International Conference on Visual Computing, Mexico City, Mexico*, Sept. 2000.
5. J. Zhao, "Embedding Robust Labels Into Images For Copyright Protection," in *Proc. of the International Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Technologies, Vienna, Austria*, August 1994.
6. J. Brassil and L. O'Gorman, "Watermarking Document Images with Bounding Box Expansion," in *Information Hiding: First International Workshop, Cambridge, U.K., May 30–June 1, 1996: proceedings*, R. Anderson, ed., *Lecture Notes in Computer Science* **1174**, pp. 227–235, Springer-Verlag, (Berlin, Germany), 1996.
7. S. H. Low, N. F. Maxemchuk, and A. P. Lapone, "Document Identification for Copyright Protection Using Centroid Detection," *IEEE Transactions on Communications* **46**, pp. 372–383, Mar. 1998.
8. S. Wolthusen, "A Model-Independent Security Architecture for Distributed Heterogeneous Systems." Unpublished.
9. C. Busch and S. Wolthusen, "Tracing Data Diffusion in Industrial Research with Robust Watermarking," in *Proceedings of the 2001 Fourth Workshop on Multimedia Signal Processing (MMSP'01), Cannes, France*, J.-L. Dugelay and K. Rose, eds., pp. 207–212, IEEE Press, Oct. 2001.
10. M. Luby, *Pseudorandomness and Cryptographic Applications*, Princeton Computer Science Notes, Princeton University Press, 41 William Street, Princeton, NJ, USA, 1996.
11. S. Wolthusen, "Enforcing Security Policies using Externally Controlled Reference Monitors." Unpublished.
12. S. Wolthusen, "Layered multipoint network defense and security policy enforcement," in *Proceedings from the Second Annual IEEE SMC Information Assurance Workshop, United States Military Academy, West Point, NY*, pp. 100–108, June 2001.
13. D. E. Bell and L. J. LaPadula, "Secure Computer System: Unified Exposition and Multics Interpretation," tech. rep., The MITRE Corporation, Bedford, MA, USA, Mar. 1976. Number ESD-TR-75-306. Performed for the Electronic Systems Division, Air Force Systems Command, United States Air Force.