Presented at the 20th National Information Systems Security Conference,
Baltimore, October 1997

A NEW STRATEGY FOR COTS IN CLASSIFIED SYSTEMS

Simon R. Wiseman, Defence Evaluation and Research Agency
and
Lt. Col. Colin J. Whittaker, UK Ministry of Defence

Abstract

The UK MOD's emerging strategy for Infosec is described. The strategy accommodates the use of modern COTS software, whilst providing security of equivalent strength to established techniques and supporting the working practices of end-users. The strategy encompasses a new approach to security policy documentation and new implementation techniques which have been shown to work with Windows NT.

1. Introduction

1.1 Background

Computer systems that handle classified information have generally been implemented in one of two ways – either as a Multi-Level Mode system or as a System High Mode system[1].

The use of Multi-Level Mode invariably leads to a requirement for security functionality in TCSEC class B1 or above, which is not available in mainstream COTS operating system products. The use of niche-market products which do provide the appropriate functionality and assurance brings a high degree of risk to a procurement, because such products are either less than fully developed or are effectively obsolete versions which do not support the latest applications.

The use of System High Mode permits the use of TCSEC C2 functionality which is readily available in mainstream COTS operating systems, such as Windows NT and Unix. However, such functionality does not permit systems operating at different system-high levels to communicate securely, because it does nothing to prevent high information flowing from the high system to the low system. Attempts to police the high-to-low flows, in order to prevent inappropriate flows, are ineffective because the information content of data is very difficult to assess, even if people (an expensive option) are used for the task.

1.2 A New Approach

The UK Ministry of Defence (MOD) recently undertook a study to ascertain a suitable procurement strategy and system architecture for its future command systems. Security apart, the study team favoured a move towards the use of COTS software and the widespread interconnection of systems. It was immediately recognised that these goals precluded both extant approaches to the construction

[1] These, and other important terms, are defined in Annex A.

of secure systems – use of System High Mode would be ruled out because of the need to interconnect high and low systems and Multi-Level Mode because of the need to use mainstream COTS software.

With no established technique seeming acceptable, the study chose to consider the problem afresh, starting from first principles, rather than postpone consideration of the problem. The proposed solution was to use interconnected Compartmented or System High Mode domains with the following security functionality:

- discretionary labelling;
- role based access controls [Ferraiolo] on shared files;
- data export between domains and compartments sanctioned by human users;
- application oriented accounting and audit.

The strategy was successful, in that the profile appeared to have the following desired properties:

- can be supported with modern COTS software;
- accommodates the work practices of the end-user;
- provides security of equivalent strength to the established techniques.

Since the proposal was for a novel approach, there was initially some reluctance to commit to it. In particular, there was no practical evidence to support the claim regarding the first two desirable properties listed above. Thus a project was started, as part of the MOD's applied research programme, to build a prototype system which became known as "Purple Penelope". This was remarkably successful in demonstrating that security need not conflict with the use of modern software and that secure systems can be usable. As a result, the new approach is rapidly gaining favour within the MOD.

1.3 Overview of the Paper

This paper describes the security architecture which has been devised, provides a security assessment of its effectiveness and gives an overview of how it can be implemented using COTS software.

The structure of existing security policy documentation, in the UK and elsewhere, is centred around descriptions of isolated systems. Modern systems, however, are invariably interconnected and it is not really possible to state the security policy for a system in isolation. Therefore, a new approach to describing security requirements and designs has been devised to accompany the new architectural strategy. This is described in section 2.

To justify the adequacy of the proposed approach, it is necessary to consider the ways in which compromises can occur, and for each determine which measures reduce the risk. It is then possible to form a judgement about whether overall risk is reduced to acceptable levels. An assessment of this form is provided in section 3.

The proposed security architecture will only be successful if it accommodates the COTS procurement philosophy. In section 4, an outline description of the Purple Penelope demonstration system is given, which is a practical example of how the proposed measures can be provided using Windows NT on the desktop.

2. Domains

2.1 The Domain Model

The new approach focuses on domains rather than systems. The intention is to capture the security aspects of the business requirements in an implementation independent way using the following model.

A Security Domain is a boundary that constrains people's freedom to move around – some people, called the domain's members, are allowed into a domain, while other people are not. Those members who are currently within a domain's boundaries are called its occupants.

Domain boundaries may be implemented through physical or logical means. An example of a domain with a physical boundary is a protected building, while a computer system may be considered a domain with a logical boundary. People enter a logical domain by "logging-in" to the computer system.

Domains are specified in order to describe the security requirements for people accessing data in repositories and exchanging data as messages. For each domain, a specification is given of which repositories can be reached by the domain's members and to which domains a channel exists for messages to be sent.

Data in a repository can only be accessed by a person while they are an occupant of a domain from which the repository is reachable. Similarly, one person can only send a message to another, if a channel exists between their two domains. These, however, are only the broad controls that are required. Further controls and restrictions will need to be imposed, and a specification of these can be made in the context of the domain model. A set of standard "security requirements classes" is being developed to aid this task.

2.2 Secure and Implementable Requirements

The domain model allows arbitrary business requirements to be specified, however not all possible requirements can be implemented securely in practice. The following simple rules are applied to a requirement, expressed in terms of domains, to confirm that the requirement is readily implementable with adequate security (as far as the UK MOD is concerned):

1. Each domain may be System High or Compartmented Mode, but not Multi-Level Mode.
2. Domains handling Secret data should contain no more than, say, 1000 members (as this is the most effective means of ensuring that the principle of need-to-know is applied).
3. No domain should have a users with clearances more than one level lower than that required to access data in any domain to which it is connected.
4. In all domains, discretionary labelling and role based access controls should be provided with E3 assurance[2].
5. Non-members to be excluded from domains through the use of physical controls and, where necessary, E3 technical mechanisms (e.g. password checks on "login").

[2] E3 is the ITSEC assurance level that is commensurate with the assurance component of TCSEC B1. ITSEC, however, has the important advantage of divorcing assurance and functionality requirements, which is vital as this strategy depends upon non-traditional security functionality.

6. Data should not be exported from one domain to another unless sanctioned by a human user, with E3 assurance that the sanction cannot be bypassed.
7. A person exporting data must be accountable for the action, with E3 assurance that accounting cannot be bypassed.
8. Communications between domains which is not required should be actively excluded through the use of E3 mechanisms.
9. Strict configuration control of software should be applied, with any proactive mechanisms being E3 assured.

A domain model is a specification which gives the implementors considerable freedom in respect of the system's design and functionality. In practice, additional requirements for security functionality will emerge as design details are taken into consideration. For example, if domains handling Secret data are implemented using public networks an encryption requirement will arise, whereas there is no such requirement if the network is private and physically protected. It is inappropriate to describe such implementation considerations in the domain model, since this is about business requirements. Instead, architectural design decisions are described using another notation which focuses on the way services are provided and used, but details of this are beyond the scope of this paper.

3. Security Assessment

3.1 The Kinds of Compromise

In order to assess the effectiveness of the approach, it is necessary to establish the different ways in which information can be compromised and consider whether each of these is countered adequately. Broadly, there are two sources of a compromise, the members of the domain and the members of any domain to which it has a connection. In all, four kinds of compromise can be identified at a high level. There are two kinds of externally sourced attack, a direct attack and an indirect attack, and there are two kinds of internally sourced attack, accidents and treachery.

In a direct attack, the external attacker breaks the domain's external communications mechanisms. Having done this, the attacker gains some freedom to observe or modify data in the domain – freedom which they are not supposed to have. For example, the attacker might find a way of observing the traffic on the domain's private network, even though they are only supposed to exchange e-mail.

In an indirect attack, the external attacker arranges that a member of the domain unwittingly causes the compromise. This may be through "social engineering" or by means of a technical attack in which the member of the domain runs software (a Trojan Horse), provided by the attacker, that carries out the inappropriate actions.

With accidents, a member of the domain makes an honest mistake. They may distribute data inappropriately or set access controls so that inappropriate access might occur. For example, a message might be sent to a distribution list which, unknown to the sender, names someone who should not see it.

With treachery, a member of the domain deliberately causes information to be compromised, and thus betrays the trust placed in them by the domain's owners. Hence, by definition, such people are traitors – regardless of whether their motive is financial reward, political gain or revenge. For example, a user may send sensitive data to their organisation's competitor for payment.

3.2 Defending Against Direct Attack

To defend against direct attack, it is necessary to ensure that only the communications services permitted by the requirement are implemented when two domains are connected. For example, the requirement may only permit the members of two domains to exchange messages, in which case it is important that a member of the external domain cannot transfer documents from the internal domain.

A firewall is a mechanism which controls communication between systems and provides confidence (formal assurance may be required, especially if the information requiring protection is particularly sensitive) that forms of communication which are not required are not provided. Provision of a firewall, however, is not sufficient. Firstly, the firewall must be constantly managed and maintained [Garfinkel&Spafford96], and secondly the configuration of the internal system must be carefully controlled, because the permitted forms of communication can be used as a carrier (called "tunnelling") for inappropriate traffic if the internal system provides suitable servers.

Thus to defend against direct attack requires both firewall and internal configuration control mechanisms to be in place.

3.3 Defending Against Indirect Attack

There is no foolproof method of preventing a Trojan Horse attack, but risk can be reduced by deploying a number of measures which make such attacks very difficult:

- detect attempts to import Trojan Horse code and reject them;
- prevent the Trojan Horse code from executing;
- limit the information which can be compromised by a single Trojan Horse;
- make it difficult for the Trojan Horse to export information;
- monitor activity to detect Trojan Horse behaviour.

A Trojan Horse may be imported as directly executable code (a binary program image), interpreted code (e.g. a macro) or a format which is compiled before execution (e.g. a Java applet). Unfortunately, distinguishing these formats from benign business data is very difficult, especially as one may be embedded in another, although it is possible to detect the signature in some cases, such as the Word templates that might carry macros or the *cafe babe* header of a Java applet [Martin&Rajagopalan97].

To prevent Trojan Horse code being executed, configuration control measures can be used. For example users may not be given the right to execute the files they create.

If a Trojan Horse manages to enter the domain and start executing, it will be working on behalf of the user who unwittingly executes the program in which it resides. This user will be constrained by role based controls and, in a compartmented domain, discretionary labelling and so the Trojan Horse is limited in what it can access, hence exposure is limited.

To compromise the confidentiality of data, the Trojan Horse needs to export it. While it is difficult to deny application software access to communications services, in many cases it is practical to ensure that their use must first be sanctioned by the human user. So, for example, when a Trojan Horse attempts to compromise information by sending an e-mail message without the user knowing, the user will be surprised by a request for export confirmation and the Trojan Horse risks discovery. To

be effective the confirmation must not be bypassable, hence a Trusted Path must be used to interact with the user. Also, the confirmation must make sense in business terms – for example the check on sending an e-mail message cannot be made at the packet level since the user would not understand whether the action is reasonable.

A final defence against an indirect attack is to monitor the system for unusual behaviour in an attempt to detect the activities of any Trojan Horse which has managed to enter the system. This is an evolving field of research [Halme&Bauer95] and products are in their infancy.

In summary, a number of measures are used to mitigate against Indirect Attacks:

- filtering imported data in firewalls and applications to prevent ingress of Trojan Horse code;
- software configuration control to prevent execution of Trojan Horse code;
- restrict scope of attack by virtue of existing constraints on the user;
- trusted path sanction of exports to hamper the Trojan Horse's attempts to leak information.

3.4 Defending Against Accidents

Documents can easily be compromised by distributing them, or their contents, to inappropriate people. Discretionary labelling provides protection against such mistakes by checking security markings against clearances. In order to gain any benefit it is important that users can easily apply an appropriate marking to their data, because markings that are too high will prevent data from being distributed in ways which are reasonable, causing operational inconvenience, whilst markings that are too low afford no protection against mistakes. It is therefore essential that users can alter markings, up and down, as they manipulate data.

The discretionary labelling checks are applied whenever data is exported. For example, when a message is sent to a distribution list, the labels of the message body and its attachments would be compared against the clearances of the recipients named in the distribution list. The message would be rejected if any recipient lacks clearance for any of the data. Thus a user who mistakenly sends information marked with a national caveat that excludes foreigners to a distribution list containing a foreign national would be saved from embarrassment.

In addition, within Compartmented Mode domains, labelling checks would be applied to shared filestores. Whenever a user attempts to observe the contents of a shared file, their clearance would first be checked against the file's label. Thus a foreign national would be prevented from observing any file marked with a national caveat that excludes foreigners, regardless of how any role based controls are set on that file.

With discretionary labelling, a user may use their discretion to give data a lower label, even if they did not originate the information it conveys. Typically this would be achieved by copying data belonging to another user and giving the copy a lower label. This is in contrast to mandatory labelling, where a user who originates some information provides a marking for it, and this marking is a mandated minimum for that information regardless of how it is subsequently used.

At first sight, it would seem that a move towards discretionary labelling rather than the more usual mandatory labelling, results in the originators of information losing the ability to control how others protect it. In practice, however, this ability is not provided even when mandatory labelling is deployed. This is because mandatory labelling is invariably accompanied by a regrading mechanism.

A regrading mechanism is essential with mandatory labelling, because such labelling has a propensity to produce overclassified data and so users need a way of adjusting the label of their own data. Unfortunately the system is not able to ascertain whether the data conveys any important information originated by other users. Accounting measures are usually associated with the use of the downgrade mechanism, but in practice these do not record the data which is downgraded and hence provide no real evidence of inappropriate downgrades.

Thus, while mandatory labelling itself does provide more protection than discretionary labelling, in practice it is coupled with other functionality that undermine its advantage.

With discretionary labelling, it is important to defend against a Trojan Horse which relabels data in inappropriate ways. Thus, when shared data is relabelled the action must be confirmed with the user using a Trusted Path. Also, when private data is exported or made shared, the user must confirm its label, to defend against a Trojan Horse which lowers the label of the data just before the user exports it.

In summary, the main defence against accidents is to provide discretionary labelling in a way which permits users to apply appropriate labels to their data with ease.

3.5 Defending Against Treachery

The main defence against treachery is the employment of personnel procedures, the provision of a good working environment which promotes loyalty and the deterrent of heavy penalties for traitors who are caught. The second defence is to limit what any one individual can do, so that if someone does turn traitor the damage is limited.

In a computer system, accounting information needs to be recorded in order to help detect traitors and, once a traitor is caught, to provide evidence and establish the extent of the damage. It is usual to rely on the operating system to collect accounting information, but unfortunately, the information collected at this level is voluminous and yet does not provide sufficient information to establish the relevant facts about the earlier actions of the system's users, such as what data they have exported. For example, a conventional "B1" operating system may record the names of files that an application reads, creates, relabels, prints and deletes, but not their contents. Consider what the following accounting information reveals:

07:25 start application WordProcessor
10:30 WordProcessor, open for read "Plans.doc", Secret
10:31 WordProcessor, close "Plans.doc"
16:50 WordProcessor, open for write "Expenses.doc", Secret
16:51 WordProcessor, close "Expenses.doc"
16:52 downgrade "Expenses.doc" to Unclassified
16:53 print "Expenses.doc" at Unclassified
16:54 delete "Expenses.doc"
18:42 quit application WordProcessor

An honest user could have performed the following actions. First they read the secret plans using their favourite word processor. Later, they used the same execution of the word processor to draft their expenses claim. This new document gets labelled Secret because the plans document previously read was Secret. Having created the expenses claim at Secret, the user adjusts the label to Unclassified, prints it and then deletes the file.

A traitor could have performed the following actions. First they read the secret plans using their favourite word processor and copy it all into the clipboard. Later that day they create a new document, paste the Secret data into it and save it as “Expenses.doc”. Having created this new document the user downgrades it to Unclassified, prints it, deletes the file. Later that night they walk out of the gate with Secret data bearing a marking of Unclassified.

Thus traditional accounting functionality is relatively poor, both at detecting traitors and establishing the damage they have caused. However, the requirement for accounting is really only important when data is passed between domains, and when it crosses compartment boundaries within a compartmented mode domain. Here, application level constraints, such as the trusted path sanction of an export, are applied in order to defend against external attacks and mistakes. Therefore, augmenting this functionality, so that meaningful application level accounting information is also retained, is relatively straightforward. For example, an archive of all printed documents and all email messages may be retained for accounting purposes.

In addition to accounting measures, it is important to place restrictions on the actions a user can perform, so that if they do turn traitor the damage they can cause is limited. Traditionally, access control lists or user/group/world access permissions have been provided by the operating system so that the users can control who has access to their data.

Unfortunately, these mechanisms are invariably difficult to apply in a fine grained way and have strange behaviour when viewed at the application level. For example, when a document is edited and saved with a modern word processor, the original file is deleted and a new file is created in its place, so any access control list attached to the document is lost.

The proposed approach is to provide users with controls which fit in with the way they use modern applications, and to present these controls in a way which makes them easy to use. Role based controls applied to directories rather than individual files would appear to satisfy these requirements. They are easier to understand, because they relate directly to physical controls such as lockable cabinets, and there is no interference from applications.

Thus the proposed defence against traitors is:

- to detect traitors and prove treacherous behaviour by gathering accounting information at the application level, which accurately and easily associates users and the information they handle;
- to deploy easy-to-use role based access controls which restrict the actions of any one user.

4. Implementing the Mechanisms

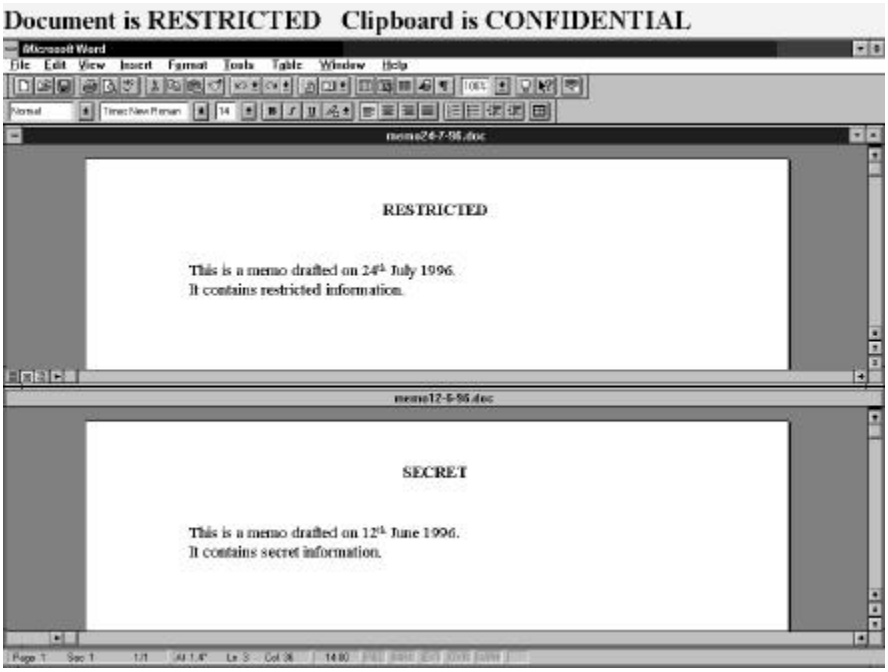
4.1 Platforms

The main platform for workstations in the near future is likely to be Windows NT, though some specialist applications may require a Unix workstation. For servers both NT and Unix will be required.

Generally, Compartmented Mode Workstations (CMWs) are not thought to have a role as workstations, though they are useful as platforms for specialist software such as a trusted DBMS. This is because many CMWs are variants of obsolete versions of Unix and hence do not support the modern applications which are required, while others are less than fully developed because the

market for them is so small. In addition, the labelling functionality provided by CMWs was intended to be used to support mandatory labelling and while, with the careful assignment of privileges, they can support discretionary labelling, the resulting system is difficult to use on the desktop. For example, the label of a document being edited cannot be lowered – it is necessary to close the document and lower the label of the file.

Although Windows NT provides E3 assured security functionality, this is of TCSEC class C2. NT does not support any form of labelling, the sanctioning of export or the high level kind of role based controls envisaged. NT does, however, have sufficient open interfaces that it is possible to tailor the use of its assured mechanisms to provide all of these mechanisms. A prototype implementation has been produced and this is the core of the Purple Penelope demonstration.



As far as the user is concerned, they see a Windows NT3.51 workstation interface, augmented with a stripe across the top of the screen in which security information is displayed. The exact content of the stripe depends on the application which is the focus of attention, but typically it shows the marking of the selected data and the marking of the data in the clipboard.

The screen shot shown above shows Microsoft Word with two documents open – one marked Secret and the other Restricted (a UK marking roughly equivalent to Official Use Only in the US). The Restricted document is current (has input focus) and so its marking is displayed in the screen stripe. Alongside is an indication that the clipboard contains Confidential data. The Secret document is visible on the screen, but its marking is only visible when the header is in view.

The system's file storage is arranged into two kinds of filestore – private and shared. Each user has their own private filestore, to which they have exclusive and unconstrained access. In addition, each user has read access to the shared filestore, subject to both discretionary labelling and role based checks.

When a file is exported (copied) to the shared filestore, the user is requested to confirm the action using a dialogue box on a trusted path (applications cannot spoof the user's input). When shared files are deleted or relabelled the action, which is subject to role based controls, is confirmed in a similar way.

Exporting a file is an accountable action, as is reading, relabelling or deleting a shared file. Shared files are never actually deleted – a copy is kept as part of the accounting information.

All files in both the private filestores and the shared filestores are labelled, and so are all running applications (processes). As applications read files, the application's label floats up according to the label of the file that is read. Similarly, when an application writes a (private) file, the file's label floats up according to the label of the application.

An application may, however, lower its label at any time, although many applications will only do this when requested to do so by the user. A common mechanism is provided through which the user can request the application to change the label of the selected data. A mouse-click on the marking displayed in the screen stripe brings up a choose-marking dialogue with which the user can select a new marking.

When an application copies data into the clipboard, the clipboard label is set to that of the application. When an application takes data from the clipboard, the application label floats according to the clipboard label. The clipboard label can be changed at any time, most easily by clicking on the clipboard marking displayed in the screen stripe. This gives the user a convenient way of extracting data which warrants a low marking from a document that overall has a high marking.

4.2 Applications

Windows applications which are run as they are shipped behave quite reasonably with respect to Purple Penelope's discretionary labelling. The limitation is with applications that handle more than one document at once, because documents open by one application are treated as having the same label. Fortunately, most such applications are also customisable by the end user, typically with plug-in modules and macros.

For Microsoft Office products, the ability to customise has been exploited to make the discretionary labelling apply at a finer granularity than the application. In the case of Word, each open document has its own label, and in the case of Access each field of a form may have its own label.

In Purple Penelope, Microsoft Access is being used to provide the forms interface for a database managed by Trusted Oracle 7 (TO7) hosted on a Sun CMW server. The TO7 DBMS has been configured so that it provides discretionary labelling, rather than the mandatory labelling for which it was designed. In addition, through the use of special database design techniques [Wiseman&Lewis95], the labelling is applied to individual attributes rather than whole entities. The user of TO7 in this way results in assured mediation at the attribute level without placing intolerable constraint on the database user.

To prepare labelled messages, with labelled attachments, Microsoft Exchange client has been customised. The current implementation, however, relies on the correct operation of Exchange and the messaging system to preserve the labels. That is, Purple Penelope currently only addresses the user interface issues of messaging, and further work is in progress to integrate this with a proper secure messaging system.

World Wide Web pages can be labelled through the use of a HTTP proxy which takes label information from the files that store pages and adds it to the HTML returned to the client. A Netscape plug-in then picks up this label information and displays it in the screen stripe.

4.3 Firewalls

The first stage of Purple Penelope aimed to show that discretionary labelling could be provided within a Compartmented Mode domain using COTS products on a single network. The second stage is underway and aims to show how such domains could be connected together.

Firewalls with E3 assured functionality are now available in the marketplace. The work currently in progress is investigating how these products can be incorporated into the domain architecture, and on how future products can be produced and evaluated quicker and more cheaply. A number of prototype simple firewalls have been produced, including one for SMTP and one for sharing files. Each is capable of providing E3 assurance against external attack, though no evaluation effort has been undertaken, and this is largely derived from the E3 assurance of a CMW. The basic technique used is that described in [Smith96].

The SMTP firewall checks the label of the message and any attachments against the clearances of the recipients. An X.500 directory has been used to provide the necessary clearance details. The label information is confirmed with the user using a Trusted Path on the user's workstation.

A "shared filestore" firewall permits domain users to export files out of the domain. From the inside, the filestore appears just like any other shared filestore in Purple Penelope, so files may be exported by drag-and-drop but each export must be confirmed using the Trusted Path. From the outside, the filestore appears as a Web site with labelled pages. Assured mediation based on labels is not, however, possible as an assured user identification service has not yet been incorporated.

5. Conclusions

The UK MOD have embarked on a new strategy for Infosec. The focus has not been on implementation techniques, nor on security policy, nor on the presentation of security policy, nor on the business needs of the end-users – instead, all these aspects have been considered together. The result is a significantly different paradigm which is unlikely to have been derived without such a revolutionary and all-embracing approach. So far it has proved very successful in exposing security issues more clearly and in permitting COTS solutions to system procurements.

6. References

D.Ferraiolo, "An Introduction to Role Based Access Control", http://waltz.ncsl.nist.gov/rbac/

S.Garfinkel&G.Spafford, "Practical UNIX and Internet Security", ISBN-1-56592-148-8, April 96.

L.R.Halme & R.K.Bauer, "Aint Misbehaving – A Taxonomy of Anti-Intrusion Techniques", Procs. 18th Natl. Inf. Sys. Security Conference, Baltimore, MD, October 1995.

D.M.Martin&S.Rajagopalan, "Blocking Java Applets at the Firewall", Procs. Internet Society Symp. on Network & Distributed Systems Security, February 1997.

R.E.Smith, “Mandatory Protection for Internet Server Software”, Procs. 12th Computer Security Applications Conference, San Diego, CA, December 1996.

S.R.Wiseman&S.R.Lewis, “Database Design with Secure DBMS Products”, Procs. 11th Computer Security Applications Conference, New Orleans, LA, December 1995.

Annex A: Terminology

A system is said to operate in **Multi-level** mode if any of its potential users have insufficient clearance to permit them routine access to some of the information which may be legitimately processed by the system. In contrast, if all users have adequate clearance for all the information, but formal restrictions apply which mean not all users are authorised to access all of the information, the system operates in **Compartmented** mode. The restrictions may be in terms of the users' nationality or the use of codewords. A system in which all users have adequate clearance and no formal restrictions apply, is said to operate in **System High** mode.

A **Security Label** is applied to a container of data to convey how the contents should be protected and distributed. A security label may be represented in a number of different ways, even within a single system. The form which is intended to be readable by humans is particularly important and is referred to as a **Security Marking**.

The functionality relating to security labels in a system may provide **Mandatory Labelling**. This means that the originator of some information is able to mandate its security marking, as represented in the computer by a security label. This marking cannot subsequently be changed by other users, regardless of how the information is used or represented within the computer. The term Mandatory Access Control is deliberately avoided because it has more than one interpretation. If MAC is interpreted as a requirement oriented statement, rather than as a mechanistic one, then it has roughly the same meaning as mandatory labelling.

Labelling may also be supported in a discretionary form, termed **Discretionary Labelling**. This is where users obtaining some information may, at their discretion, change its marking. Typically, the change is made as data conveying the information is copied.

A **trusted path** is an interface between two parties which assures one party of the identity of the other. Often, the assurance of identity is mutual. The most common trusted path is that used between the human user and the function which checks or changes their password.