Defence Research and Development Canada
Recherche et développement pour la défense Canada

Installing and configuring a declassification system

A solution combining Trusted Solaris and Free and Open Source Software

Richard Carbone
Certified Hacking Forensic Investigator (EC-Council CHFI)
Certified Incident Handler (SANS)
DRDC Valcartier

Simon Vincent
Collège O'Sullivan

Defence R&D Canada – Valcartier

Technical Memorandum
DRDC Valcartier TM 2009-086
February 2013

Installing and configuring a declassification system

A solution combining Trusted Solaris and Free and Open Source Software

Richard Carbone
Certified Hacking Forensic Investigator (EC-Council CHFI)
Certified Incident Handler (SANS)
DRDC Valcartier

Simon Vincent
Collège O'Sullivan

Defence R&D Canada – Valcartier

Technical Memorandum
DRDC Valcartier TM 2009-086
February 2013

Principal Author

Original signed by Richard Carbone

Richard Carbone

Programmer/Analyst

Approved by

Original Signed by Guy Turcotte

Guy Turcotte

Head/Mission Critical Cyber Security Section

Approved for release by

Original signed by Christian Carrier

Christian Carrier

Chief Scientist

Abstract

In spring 2008, certain project requirements necessitated the use and implementation of a system that could be used to declassify classified data for use on unclassified systems and networks. Some of the researchers at Defence Research Development Canada – Valcartier wished to be able to share declassified versions of their work with others. However, data residing on classified systems and networks cannot easily be transmitted to unclassified systems. Although it can be done, a variety of safeguards, data containment and compartmentalization and various procedures must be followed for doing so. The challenge is not so much the procedures and safeguards, but rather using an accredited system for containing and compartmentalizing the data to be declassified. Since no DND policy is clear on which systems to use or the process by which data can be declassified, the primary author decided that by combining commercially available software with specific FOSS components, a secure system could be built that would satisfy even the most stringent security requirements. The solution chosen was a combination of Trusted Solaris and various commonly used FOSS packages. This memorandum therefore describes how such a system is built and configured.

Résumé

Au printemps 2008, certaines exigences de projets nécessitaient l'utilisation et l'implantation d'un système qui pourrait être utilisé pour déclassifier des données classifiées pour une utilisation sur des systèmes et des réseaux non classifiés. Certains chercheurs à Recherche et développement pour la défense Canada – Valcartier voulaient être en mesure de partager des versions déclassifiées de leurs travaux avec d'autres. Toutefois, des données résidant sur des systèmes et réseaux classifiés ne peuvent pas facilement être transmises sur des systèmes non classifiés. Même si cela peut être fait, un éventail de protections ainsi que différentes procédures pour le confinement et le cloisonnement des données, entre autre, doivent être suivies pour y parvenir. Le défi n'est pas tant les procédures et les protections, mais plutôt l'utilisation d'un système accrédité pour le confinement et le cloisonnement des données à être déclassifiées. Puisqu'aucune politique du MDN est claire sur les systèmes à utiliser ou le processus par lequel les données peuvent être déclassifiées, l'auteur principal décida qu'en combinant des logiciels commerciaux disponibles avec des composants à code source libre et ouvert, un système sécuritaire qui satisferait même les exigences de sécurité les plus strictes pourrait être construit. La solution choisie est une combinaison de Trusted Solaris et de différents progiciels à code source libre et ouvert couramment utilisés. Ce mémorandum décrit donc comment un tel système est construit et configuré.

This page intentionally left blank.

Executive summary

Installing and configuring a declassification system: A solution combining Trusted Solaris and Free and Open Source Software

Carbone, R.; DRDC Valcartier TM 2009-086; Defence R&D Canada – Valcartier; February 2013.

Specific classified projects require that eventually, data and results obtained from systems specific to one or more classified networks be shared and disseminated with colleagues and various collaborators. However, declassifying data is not an easy task. Unless there exist two simultaneous versions of an experiment, each conducted independently of the other (one classified and the other unclassified), results from the classified experiment cannot generally be released, as it would be deemed to derive, directly or indirectly, from a classified project. Of course, certain classified specifics can always be generalized to the point that the data can be readily declassified.

However, it is not only experiments and their results that require occasional declassification. Other instances may necessitate those different types of files detailing various specifics, whether about a project, experiment, research or collaboration, to be shared and disseminated with others. The problem in so doing is that often individuals wishing to receive a copy of the classified file(s) either do not have adequate access or the files are deemed "need to know", thereby restricting the files, their data and potential audience. However, if the files can be adequately stripped of details and information concerning highly classified materials and other "need to know only" material, then the files can be disseminated across a larger community. This dissemination enables researchers to further share and propagates their ideas, technologies and results. This larger audience may be able to provide additional contributions or refinements to the shared work they received. Of course, those in this larger community must have adequate security clearances so that they can work on and access the material in question.

The problem of data declassification is not a new one, as it is has persisted and plagued security researchers, system administrators and information security officers for many years. However, systems already exist which are fully capable of handling, storing and processing data at multiple levels of classification. These systems are commonly referred to as MLS (multi-level security) systems and are used extensively by military and intelligence agencies throughout the world for treating and working with classified electronic information. Some of these systems have obtained TCSEC (and where applicable Common Criteria) accreditation; as such, some of can be readily accepted in high-security environments, as approved by their national accreditation agency.

This memorandum will therefore demonstrate how a readily procured commercial MLS system, Trusted Solaris 8 (TSOL), when combined with appropriate open source software, can be used for editing and stripping out classified material from a file which, upon approval, can be downgraded and disseminated at a lower level of classification. However, this memorandum does not assume that highly classified data can simply, upon editing, be downgraded to a publicly releasable version. On the contrary, it is assumed that a file will be undergoing appropriate modification to go from highly classified to a lower level of classification. However, the original file nevertheless remains classified.

Sommaire

Installing and configuring a declassification system: A solution combining Trusted Solaris and Free and Open Source Software

Carbone, R. ; DRDC Valcartier TM 2009-086 ; R & D pour la défense Canada – Valcartier; février 2013.

Certains projets classifiés particuliers exigent qu'éventuellement, les données et les résultats obtenus à partir de systèmes spécifiques à un ou plusieurs réseaux classifiés, soient partagés et diffusés avec plusieurs collègues et collaborateurs. Toutefois, déclassifier des données n'est pas une tâche facile. À moins qu'il existe deux versions d'une même expérience, chacune menée de façon indépendante l'une de l'autre (l'une classifiée et l'autre pas), les résultats de l'expérience classifiée ne peuvent généralement pas être communiqués, car ils seraient considérés comme dérivant, directement ou indirectement, d'un projet classifié. Bien sûr, certaines particularités peuvent toujours être généralisées au point où les données peuvent être facilement déclassifiées.

Cependant, ce ne sont pas seulement des expériences et leurs résultats qui exigent une déclassification occasionnelle. D'autres cas peuvent nécessiter que différents types de fichiers, fournissant des détails, que ce soit à propos d'un projet, d'une expérience, d'une recherche ou d'une collaboration, doivent être partagés et diffusés avec d'autres. Le problème, ce faisant, est que souvent, les personnes qui souhaitent recevoir une copie du ou des dossiers classifié(s) n'ont soit pas un accès adéquat ou les fichiers ont une nécessité d'accès, limitant ainsi leur public potentiel. Cependant, si les fichiers peuvent être adéquatement dépouillés des détails et des informations concernant du matériel hautement classifié ou avec une nécessité d'accès, ils peuvent être diffusés à une communauté élargie. Cette diffusion permet aux chercheurs de partager et de propager davantage leurs idées, leurs technologies et leurs résultats. Ce public élargi peut alors être en mesure de fournir des contributions supplémentaires ou des améliorations aux travaux ainsi reçus. Évidemment, les personnes de cette communauté élargie doivent avoir les attestations de sécurité adéquates afin qu'elles puissent travailler et accéder au matériel en question.

Le problème du déclassement de données n'est pas nouveau, car il perdure et afflige les chercheurs en sécurité, les administrateurs de systèmes et les officiers de sécurité depuis de nombreuses années. Cependant, des systèmes existent déjà et sont pleinement en mesure de gérer, de stocker et de traiter des données à plusieurs niveaux de classification. Ceux-ci sont communément appelés des systèmes à niveaux de sécurité multiples et sont largement utilisés par les agences militaires et de renseignement à travers le monde pour traiter et travailler avec des informations électroniques classifiées. Certains de ces systèmes ont obtenu l'accréditation TSEC (et le cas échéant, des Critères communs). Quelques-uns peuvent être facilement acceptés en tant que tel dans des environnements à haute sécurité, étant approuvés par leur agence nationale d'accréditation.

Ce mémorandum démontra donc comment un système MLS commercial aisément acquis, Trusted Solaris 8 (TSOL), lorsque combiné à des logiciels libres appropriés, peut être utilisé pour éditer et dépouiller un fichier de tout matériel classifié qui, apprès approbation, peut être déclassé et diffusé à un niveau inférieur de classification. Toutefois, le présent mémorandum ne présume pas

que des données hautement classifiés peuvent tout simplement, après édition, être déclassées à une version qui peut être rendue publique. Bien au contraire, on suppose qu'un fichier fera l'objet d'une modification appropriée pour passer d'un niveau très élevé à un niveau inférieur de classification. Toutefois, le fichier d'origine reste néanmoins classifié.

Table of contents

Acknowledgements

The author would like to acknowledge and is very grateful for the major contributions provided by Simon Vincent, who worked under the author throughout the months of June and July 2008 as a coop student. Without Mr. Vincent's hard work, this memorandum would not have been possible. The author would also like to thank Mr. Yves van Chestein for reviewing this document and providing comments and corrections for it in its Technical Note incarnation.

The primary author would also like to extend a large thanks to Philippe Charland who helped provide the necessary comments and corrections in order to reincarnate this document as a Technical Memorandum. Moreover, the primary author would also to thank him for peer reviewing this document.

This page intentionally left blank.

1 Introduction

1.1 Objectives

This memorandum aims to provide basic and fundamental information to the reader pertaining to operating systems security. More specifically, it examines the fundamentals of maintaining and enforcing system access control, securing operating system features, followed by a discussion exploring why such systems are sometimes necessary. Furthermore, a brief examination of the certification and accreditation process is also provided. These portions are provided in Section 1 of this providing comments and corrections.

In Section 2, the installation and basic configuration of a Sun Microsystems Trusted Solaris environment are examined. This includes the overall installation procedure as well as any specific environmental modifications.

Annex A examines how data files and documents are downgraded so that they can be burned to CD or DVD-based media. Annex B examines how data files, no longer required as they have been transferred to removable media, and are securely deleted. Annex C describes other important issues including the use of various anti-virus and rootkit scanning tools as well as maintaining an appropriate update policy for a TSOL 8 system.

This memorandum was written to address the lack of available information available to DRDC and to the public in general and indicate how to use a secure operating system for modifying, downgrading and transferring classified materials to lower levels of classification. Currently, no publicly available source of information details how to proceed with data declassification. Furthermore, no current documentation exists for carrying out such operations using a secured operating system. The reason for using a secured operating system is that Windows-based systems cannot be adequately trusted to safeguard and compartmentalize highly classified information and enforce a need-to-know and need-to-access environment.

This memorandum is not a UNIX tutorial nor should it be construed as one. Furthermore, it is not meant to teach readers how to appropriately edit and remove classified documents for downgrading and eventual declassification. Instead, the reader should already be familiar with these techniques. Finally, the reader must respect his institution or government department's policies regarding document declassification.

1.2 Requirements

This memorandum presupposes that the reader has had some UNIX experience, preferably with the Solaris environment. Although it is not necessary for the reader to have used a secure operating system before, such as Sun Microsystems Trusted Solaris, it is preferable. Many of the operations required to install and configure a Trusted Solaris environment are not specifically examined; however, ample documentation is available from Sun's Solaris documentation web portal.

1.3 Multi-level security

Multi-Level Security, commonly known as MLS, is a unique method for ascribing specific security levels and labels to a computer operating system, thereby enabling data compartmentalization. More specifically, through a series of well-defined security levels and appropriately labelled data files, directories and processes, a computer system can readily segregate all of a user's data, information and processes from all others on the system, including the system administrator. Furthermore, a user's own data and information can be compartmentalized so that his various data can reside at different levels of security, thereby ensuring that data from one security level does not cross-pollinate data on a different security level. Security levels are also commonly referred to as classification levels. In addition, MLS-based systems permit the configuration and fine-tuning of security levels, access controls and allowable actions and privileges for both users and system administrators. [1, 2]

MLS is generally accomplished by implementing a form of MAC (Mandatory Access Control). This ensures that the necessary logical system structures, permitting both data and processes to run and exist at different security levels, all while segregating and compartmentalizing both data and information, have been correctly implemented [1, 2, and 16]. These implementations occur at both at the system and user level. This is sometimes referred to as compartmentalization and polyinstantiation. MAC-based MLS is an implementation of the Bell-LaPadula Model, which formally defines the United States' Department of Defence MLS policy [1, 2, 4 and 16].

MLS-based systems are commonly known as trusted operating systems (TOS) and are generally only available commercially [1, 2 and 54]. Examples of commercial TOS systems include Sun Microsystems' Trusted Solaris (TSOL) [3, 5, 38, 39, 40, 41, 42 and 47], Hewlett-Packard's Trusted HP-UX [6] and SGI's Trusted IRIX (TRIX) [7, 21, 34 and 35]. Other systems exist, although less popular, including IBM's Trusted AIX platform [8]. All of the aforementioned systems were at one time Trusted Computer System Evaluation Criteria (TCSEC) certified [45]. Today, those that are still in use today (excluding Trusted HP-UX) have received Common Criteria[1] (CC) certification [44].

Currently, there are several ongoing open source MLS initiatives including SELinux (Security-Enhanced Linux), SEBSD[2] (Security-Enhanced BSD) and FreeBSD [1, 2, 9 and 10]. Although all three systems implement MAC-based MLS policy-driven functionality, they do not incorporate all of the same characteristics as their commercial counterparts [1, 2 and 16]. As good and as widely used as SELinux [9, 50, 52, 53, 58 and 59], SEBSD and FreeBSD [10, 51, 56 and 57] systems are, they are currently not up to par with TSOL [3, 5, 38, 39, 40, 41, 42 and 47] or TRIX [7, 21, 34 and 35].

Thus, although the aforementioned open source systems may meet basic[3] MLS requirements, they should not be considered as robust as their commercial counterparts. The vast majority of commercial TOS systems were built according to DoD's TCSEC specification [25, 30]. It is for this reason, although some may believe it to be incorrect, that commercial TOS systems are

[1] Common Criteria certification has superseded TCSEC certification. At this time, all TCSEC certification efforts have ceased.
[2] SEBSD is also commonly known as TrustedBSD.
[3] This is according to specific certification-based methodologies such as CC [28, 46, 73 and 74].

generally more secure than their FOSS counterparts, even though FOSS-based systems are open to source code verification and analysis [26, 27]. However, this does not preclude these systems from eventually attaining the same level of maturity and security as their commercial counterparts.

1.4 Security differences between commercial and FOSS-based TOS systems

Commercial and FOSS-based TOS systems differ on a variety of issues. Primarily though, they differ due to distinct differences between their security frameworks and architectures. Although these systems rely on various implementations of MAC for implementing system security, commercial systems generally offer increased security capabilities by integrating additional technologies into their TOS systems. For example, comparing TSOL [3, 11, 38, 39, 40, 41, 42 and 47] with SELinux [9, 50, 52, 53, 58, 59, 60 and 61], SEBSD and FreeBSD [10, 49, 51, 56 and 57], it is easy to understand how TSOL, at least to date, is a superior and more secure product as compared to either FOSS alternative. However, this is not to say that SELinux, SEBSD and FreeBSD are lesser products, only that they are less mature and, due to their immaturity and lack of configuration tools, are likely to lead to systems being incorrectly configured, resulting in an overall less secure system.

Moreover, systems including SELinux, SEBSD and FreeBSD are capable of implementing the vast majority of security features found in TSOL, due to their implementation of MAC-based Type Enforcement (TE) [55]. However, a great deal of work is required to adequately and securely configure them [9, 10, 51, 58, 60 and 61]. Currently, most configuration efforts must be done manually. Even with the introduction of several new GUI security policy-based configuration tools [58], the underlying system itself does not adequately implement an overall system security policy [9, 10]. However, this does not preclude an individual from creating one [9, 10, 58 and 60]. Moreover, it will not be possible to create an easily modified generalized policy that can be used for the vast majority of cases. This explains the necessity for creating application specific security policies.

When compared to TSOL [3, 39, 40, 41, 42 and 47] however, these FOSS-based systems lack advanced system management tools, policy and privilege editors and easily modifiable security configuration files [9, 10]. However, these FOSS systems have one unique advantage, which is that all the source code is freely available for modification and analysis. In addition, much technical documentation exists for SELinux [9, 50, 52, 53, 58, 59 and 60], SEBSD and FreeBSD [10, 51, 56 and 57], which is not available for commercial systems including TSOL and TRIX.

When comparing well-established TOS systems including TSOL [3, 38, 39, 40, 41, 42 and 47] and TRIX [21, 34 and 35] with FOSS systems, the reader may develop a slight bias against the latter. The fact is that the primary author has high praise for FOSS and is only highlighting issues that must be worked on in order to improve them so that they too can be made easier to use and administrate. It is a well-known fact that SELinux, SEBSD and FreeBSD systems require highly complex TE rules and policies [9, 10, 58 and 60], some of which can span many hundreds of lines in order to adequately secure a given application. Little work has currently been done by the FOSS community to improve the management and application of TE rules and security policies.

Locking down a SELinux system for use in a highly classified environment where all user processes, data and devices must be compartmentalized and segregated can be accomplished. However, default system configuration policies for SELinux, SEBSD and FreeBSD do not make it easy to implement. It is very likely that configuring either system appropriately would take more time than learning how to implement and put TSOL or TRIX into place. However, the cost of TSOL or TRIX can be prohibitive, depending on underlying requirements. Importantly, TSOL runs under both SPARC and Intel x86 architectures, making its implementation easier than that of TRIX. Unfortunately, most SGI workstations have rather prohibitive costs whereas Sun Microsystems workstations are relatively inexpensive[4]. Moreover, both TOS systems have appropriate system security configuration policies that can be implemented and rolled out at once, although the fine-tuning of specific parameters and performance can always be adjusted later on.

Various studies have shown that in general, mature FOSS projects are at least as secure as their commercial counterparts, if not more so [62, 63]. Although the source code for SELinux, SEBSD and FreeBSD is fully available for comparative studies, the source code of commercial TOS systems is not. Therefore, comparative correlations between commercial and FOSS TOS-based systems will have to remain qualitative in nature, based on results pertaining to system performance, system administration, security configurability, functionality and usability.

The unique advantage of FOSS over commercial systems where source code is not available is that it is possible to perform in-depth source code analyses. With appropriate training and tools, it is possible to carryout detailed source code audits of any FOSS-based system [26, 27], including the kernel as based on recommendations from various software and best coding practices[5] [13], return to the code and then make the necessary changes. With commercial systems where only binary executables are available, reverse engineering is too complex[6] to be readily feasible.

It cannot be determined if the source code to commercial TOS systems has undergone rigorous screening and whether or not they are based on the same code as their unsecured counterparts. SELinux, SEBSD and FreeBSD are generally based on the same code base as their unsecured counterparts, with the exception of the necessary kernel modules and administrative tools and policies.

1.5 Expectations of an MLS system

Although all MLS-based systems differ slightly from one to another, they all provide the same basic features. These features are difficult to rank in terms of their overall importance, as they are

[4] The build quality between Sun and SGI workstations, which could partly explain the price difference, is contentious and cannot be resolved here.
[5] Consider GCC ProPolice (http://www.trl.ibm.com/projects/security/ssp/) [12, 72] and David Wheeler's Secure Programming for Linux and Unix HOWTO [13]. The former provides some protection against stack smashing and buffer overflows. This is accomplished by compiling in this protection mechanism at compile time while the latter provides best practices for securely programming in C/C++ under UNIX/Linux.
[6] Reverse engineering software is by all accounts illegal without the explicit permission of the author(s) of the software in question. Before attempting to reverse engineer any piece of copyrighted software, consult with local management and/or legal counsel.

all critical to the secure well-being of a given computer system. Most MLS-based systems, regardless of vendor-specific MLS security mechanisms, all generally provide the same overall security functionality, in so long as the security mechanisms are consistent with TCSEC. [1, 2 and 30]

The following is a short, concise list of the various features found amongst the aforementioned commercial MLS systems (TSOL, TRIX, Trusted HP-UX and Trusted AIX). Descriptions of these features can be found at the end of this memorandum in the Glossary section.

- Mandatory Access Control [16],
- Discretionary Access Control [19],
- Access Control Lists [18],
- Role-Based Access Control [17],
- Auditing and Accounting [14],
- Device Allocation [3],
- Privilege Escalation [3],
- Principle of Least Privilege [20]
- Polyinstantiation [3],
- Trusted Path [15, 75],
- Trusted Networking [48].

In general, it is expected that an MLS-based system will be secure, e.g., it will be able to adequately compartmentalize data, information, processes and various accesses (including the network). Furthermore, all accesses, whether user or system administrator-based, must be on a need-to-know, need-to-access or need-to-modify (system administrator only) basis. Moreover, even if an attack or exploit is successful, the system must sufficiently prevent an attacker from gaining high-level system access. In an MLS system, this is accomplished through various safeguards, including the assignation of duties through roles and profiles, compartmentalization of data and resources through MAC and privilege escalation. Insofar as a system administration role or profile is successfully compromised, then this role/profile must be limited with respect to what it can do to the system. [1, 2, 3, 16, 17, 19, 20, 30 and 54]

Under an MLS environment, normally a user cannot gain access to other users' data, unless it has been shared using DAC and it is at the same security level as the current user's session [1, 2, 3, 16 and 19]. Furthermore, system administration roles and profiles cannot ordinarily access user data or contrariwise [16, 17 and 19]. Moreover, only authorized and assigned users should be able to access key roles or profiles in order to carry out system-specific administrative tasks (RBAC) [17]. In addition, users cannot access the data or processes of roles or profiles and contrariwise [1, 2, 3, 16 and 17]. In short, the system must adequately segregate and compartmentalize both users and system roles or profiles.

As with any complex computer operating system, it is expected that mechanisms will exist to perform the fine-tuning of system permissions and privileges [1, 2, 16, 17, 21, 30, 34, 35 and 47].

However, this is not always the case and often, depending on the system, changes must be made to the system manually, even if basic support tools exist (e.g. SELinux, SEBSD and FreeBSD) [9, 10, 49, 50, 51, 52, 53, 56, 57, 58, 59 and 60]. It is, however, the objective of this memorandum to carry out a comparative analysis between commercially available and FOSS-based TOS systems. Nevertheless, all TOS systems provide some mechanism, even if it is as arbitrary as modifying text configuration files, for assigning and delegating permissions and privileges and compartmentalizing data, processes and resources from one another.

Overall security is maintained by assigning users, roles, profiles and other system objects only enough permissions and privileges so that they can carry out their specific activities and duties [17, 20]. MAC provides MLS functionality by supplying the necessary logical framework to the system defining how objects interact with one another and what data or information can be interchanged between them, by assigning to the various system objects varying security contexts and labels [1, 16]. Under UNIX, every system component is an object. These include devices, processes, applications, libraries and users, all of which can be assigned a security context and label. However, an object also requires various permissions and privileges in order to correctly interact with the system [20]. This can include interacting with other users, devices, processes, applications and data. Therefore, when different objects interact with one another, the overall security context of interacting objects must be respected by the system, including providing a requesting object with the least amount of data or information necessary for it to complete its functions and duties. Of course, the system administrator can always fine-tune the security context of any object. However, great care must be taken not to break overall system security by assigning any one object too many permissions and privileges such that it can override the underlying security afforded by the various security mechanisms. Doing otherwise will break the default configuration, thereby invalidating the operating system's pre-existing accreditation. [15, 18 and 75]

Users can generally share data with one another and with other objects (e.g. applications, web services, databases, etc.). However, this is possible only if the object accessing the shared object has the same security context as the shared object. The same is true for system roles and profiles. They too can share data with one another, assuming they are running in the same security context. However, under no circumstances, unless configured otherwise, should user objects be able to access information from roles and profiles and contrariwise. Furthermore, due to MAC, if an object is running at a higher security context than a given data file (or other object), the object can generally read and write to that file, assuming proper read/write permissions have been assigned. However, an object running at a lower context than another given object cannot and will not be able to read or write to that object, even if appropriate permissions have been assigned. This is one functional aspect of MAC and is known as Write-Up-Read-Down (WURD). Additionally, even if privileges have been assigned to an object, normally WURD is respected, unless key privileges are assigned to an object by the security administrator. This, however, breaks all TCSEC and CC certification and accreditation standards. [1, 16, 19, 20]

As serious as TOS systems take security, no system is foolproof or bulletproof. Bugs will always exist in software, especially when written in lower-level programming languages such as C, which accounts for more than 95% of all UNIX-based operating system code. However, where security is a concern, information and data segregation and compartmentalization implemented by a hardened operating system kernel can make a great deal of difference in securing important data and keeping it restricted. TOS systems are highly utilized in classified environments, although

Windows-based systems have been making inroads. Windows systems though, do not implement anywhere near as many security features as TOS systems. Unfortunately, all the additional security features of TOS systems add further administrative complexity on top of an already difficult system to manage.

1.6 Necessity of MLS systems

MLS systems are often considered unavoidable by those working in IT security. They are at times a required nuisance for working with and processing classified or sensitive information [38]. However, to use an unsecured system in lieu is to put security-critical data at high risk. The entire purpose of using an MLS-based system is to secure and safeguard important information such as industrial trade secrets, protect classified information and maintain confidentiality (e.g. client banking information [76]).

Generally, MLS systems are not connected to wide-area networks such as the Internet due to the nature of the data they store and process, but nothing prevents this situation from occurring. Nonetheless, it is not a good security practice to connect systems with classified or restricted data directly to public systems. Instead, these systems should be connected to “discrete” or isolated networks to avoid the potential for data being gleaned from them through user-based error or successful vulnerability penetration. This is particularly true for military and intelligence-based organizations, where critical information is kept on isolated networks in specialized and secured facilities so that only authorized users have access to both the systems and the room there are located within.

Many institutional and commercial entities also use MLS systems but they do not generally advertise that fact. Instead, these organizations tend to be wary about the type and amount of information they release to the public concerning their computer systems. It is certainly a well-known fact that institutional clients such as banks deploy MLS systems as web-based portals used in processing online business transactions. MLS systems offer information and access-based protection mechanisms not commonly found in popular computer operating systems such as Windows.

The banking industry is notorious for its propensity for using mainframes. Mainframes make up a bank’s core datacenter. Furthermore, these systems are almost never directly connected to public networks. Instead, satellite systems such as MLS web portals conduct the business transactions in tandem with the core systems, where they act as secured relays between a public system (e.g. client’s remote Internet-connected computer) and the core systems (e.g. mainframe). MLS systems offer security advantages that hinder an attacker’s ability to break into the system and install malware that can be used for snooping on client information. Successful break-ins are normally mitigated by the restricted access the system places on accounts and roles. Known vulnerabilities are commonly exploited by hackers to break into various commercial entities. It is obvious that all computer systems, even MLS MAC-enabled certified commercial systems, have vulnerabilities. While MLS systems do not guarantee that the underlying systems are error free, they do instead place great emphasis on mitigating the access to various system resources that may be successfully broken into in order to bypass system security. [1, 2, 3, 16, 17, 19, 20, 30, 54]

However, if it becomes necessary, MLS systems can be connected to any standard network, such as Ethernet, Token-ring, ATM, given that adequate safeguards are taken, including heavy data encryption. The networking requirements and the protocols supported by MLS systems are generally no more different from those of their non-MLS counterparts. However, the majority of MLS systems do support enhanced security telecommunication protocols for the exchange of network data at different levels of classification using CIPSO and RIPSO, both of which run over TCP/IP. Using either of these protocols, trusted systems, even if they are from different vendors, can exchange labelled information with one another. [3, 22, 23, 25, 29, 30, 39, 40, 69 and 70]

Unfortunately, MLS systems are far more complex than non-MLS systems, to both configure and use on a daily basis. They are highly burdensome for the system administrators, who must contend with them and users find their use irksome at best. However, there are valid instances when these systems must be used. Nevertheless, these systems are complex to use, manage, administrate, configure and maintain. This requires that both the system administrators and users be advanced in their system management and system use capabilities, respectively. Anyone who has ever managed or even tried to configure an MLS system understands the involved complexities. To the uninitiated, this may seem trivial, until they experience first-hand the difficulties it involves.

1.7 Certification and accreditation

Although operating system certification accreditation are not absolutely necessary for TCSEC-compliant systems [25, 28, 73 and 74] including TSOL [3, 5 and 47], TRIX [7, 21], Trusted HP-UX [6] and Trusted AIX [8], it can provide great comfort to clients knowing that a given system has been formally tested by competent testing laboratories. Unfortunately, a complete listing of all secured TCSEC-compliant operating systems is no longer available from the NSA or NIST as the list has been superseded by CC certified and accredited systems [44]. The American TCSEC evaluated product list is no longer available. Instead, a nearly complete list is still available from the Australian government [45].

It is the author's belief that the CC specification is nowhere near as stringent as the TCSEC specification. An in-depth comparison between both specifications reveals this fact. However, the process by which testing and accreditation occur are better explained and outlined under the CC specification [25, 28, 30, 46, 73 and 74]. Many could argue, however, that the CC specification is a more pragmatic document against which to test and build TOS systems. It is the author's opinion that FOSS-based systems, due to the way they are built and worked upon by the community at large, benefit more from a simpler and more concise security framework as put forward by the CC specification [25, 28, 73 and 74]. However, the accreditation process for FOSS is complex and expensive and can only move forward if a commercial entity can pay the expenses as the open community by itself does not have the necessary financial resources. This cost easily rises into the many hundreds of thousands and sometimes millions of dollars.

Unfortunately, FOSS-based systems that do implement MAC-based security mechanisms systems do not yet incorporate the full array of security features expected of fully compliant TCSEC operating systems. The vast majority of commercially available MLS systems are B1, B2 or B3 compliant with the TCSEC standard [30], at least until recently. Today, the CC specification has become the industry accepted standard and has replaced TCSEC and equivalent standards. The author is of the opinion that the CC specification is not as stringent as TCSEC. In addition, the author posits that since FOSS-based operating systems are today measured against the CC specification, they are unlikely to be as secure as their commercial counterparts that at one time were generally TCSEC level B compliant.

It is likely that at one point, FOSS-based MAC-enforcing operating systems will reach the same level of security as their commercial counterparts. This may eventually be achieved through enough subsequent source code audits that the code can be considered as highly reliable and secure or that the MAC-based capabilities of these systems have matured sufficiently to make up for other shortcomings. Currently, SELinux and its community have to take this path. In so doing, the vast majority of capabilities enjoyed by other MLS TCSEC certified systems would also be enjoyed by SELinux, as its implementation of MAC will effectively lock down the system in a similarly effective manner. Of course, it will not be the same and will still be lacking certain capabilities including polyinstantiation and Trusted Path-based capabilities [3].

It is hoped that in the future, advanced programming language paradigms such as AOP (Aspect-Oriented Programming) will enable advanced security features to be integrated directly into software at compile time by integrating and weaving a security policy into the source code [14]. However, it is not yet known exactly how effective these programming languages have been in diminishing the effectiveness of malware and the rise in computer-based attacks. Time will tell when meaningful studies and metrics have been established for these programming paradigms.

All software used by the Canadian government is certified by CSEC (Communications Security Establishment Canada), which has the responsibility of ensuring that government used software is safe and robust. CSEC is Canada's certifying authority for CC and CTCPEC compliant computer and software systems. CTCPEC was put forward in 1993 by the CSEC as Canada established its own trusted computing guidelines [37]. In the US, the NSA/NIST is the official American certifying authority.

1.8 Declassification and sanitization

There are varieties of reasons why data may be declassified [65]. Most often, it is due to the need to share information with others, either with colleagues, partners or members of the public under a request based on the Access to Information Act [66, 67]. Classified information may also be requested by other government departments or from within the current department with requests originating from various levels of departmental management. In any case, data, images, records and other critical information contained within these restricted data files at high levels of classification (e.g. Secret (S), Top Secret (TS)) must be screened and purged before they can be disseminated to those lacking appropriate clearance. Even in the case where other government colleagues have appropriate clearance, there must be a valid reason for requesting classified information as it is generally made available only on a need-to-know basis. The same principle also applies to departmental management. Therefore, it is vitally important that before any classified information is screened, purged and disseminated that appropriate channels are used and that the requesting party's security clearance credentials have been verified. The local Information Systems Security Officer (ISSO) can be of assistance in determining and understanding the various rules and procedures involved in information declassification.

Once approval has been obtained, data can then be declassified to the appropriate level of dissemination. If the classified data in question is considered as Top Secret (TS) and it is to be shared with colleagues having only Secret (S) level clearances, then any TS-level information must be expunged from the data files and documents. Even more worrisome than sharing information with colleagues is sharing it with the public under an Access to Information request. In such a case, all classified, confidential and sensitive information must be permanently expunged before any public dissemination is possible. At all times the public should be considered unscreened and non-cleared for handling any form of classified, confidential or sensitive information.

Therefore, declassification consists of downgrading data files and documents to the point where any information for which one or more individuals are not cleared cannot be accessed, obtained or construed from a given data file or document. It is important to consider that declassification procedures vary from site to site and from department to department. In addition, it is important that the reader be familiar with his own departmental procedures as the violation of security rules concerning the declassification of highly classified information can be considered not only a breach of national security, but may also be a punishable criminal offense [64]. The act of expunging data or records from data files and documents is known as sanitization [24] and is a key component for declassification.

It is extremely important that the original author of the document, files or data declassify the requested information. This is because only the author knows the information contained within a given file, data, or document and is therefore the authority on which information constitutes a given level of classification. Furthermore, declassification of data files and documents by anyone other than the original author is generally strictly prohibited unless management has given express consent to do otherwise.

Once approval for declassification has been obtained following appropriate channels, the data must be retrieved from the classified system. The classified system itself may be connected to a classified network or maybe standalone in nature. In either case, because the declassification system is also standalone, data must be transferred from the originating classified system to the declassification system. Data can be transferred using a variety of means, including CD or DVD, USB memory stick[7], floppy disks or other removable disk-based devices (e.g. ZIP drive, etc.). Under this proposed methodology, the declassification system must remain isolated and is not to be connected to the network. Data is transferred by the user (the author) from the originating classified system to a compatible device type that the declassification system will recognize (most likely a CD or DVD). This device is then to be connected to the declassification system by the system or security administrator who will then grant the user the appropriate rights and privileges to copy the data to his system work environment. Because the declassification system is a MLS-based system, ordinary system users are not authorized to transfer data from non-work environment devices including CDs, DVDs and other removable devices. Upon completing the data transfer, the user's data transfer-required rights and privileges must be revoked in order to prevent any potential breach in security. These rights and privileges can only be attributed to users by the system or security administrator of the declassification system. In addition, any data transfer media (e.g. CD, DVD, etc.) used to transfer data between the system of origin and the

[7] Use of USB devices may have been authorized in specific locations. Check with the local ISSO to be certain whether they are permitted.

declassification system must be securely locked up by the system or security administrator or other individual responsible for media security in order to prevent the misappropriation of classified materials.

The user's (author) system work directory on the declassification system generally denotes the user's home directory. It is important that the transfer data, while it resides on the transfer media, is unlikely to have any specific MLS-based security or classification label since the originating systems are usually Windows-based and these systems do not support security or classification labels. Therefore, to preserve the integrity in the chain of data classification, the user must login to the declassification system using the same level of clearance the transfer data currently resides at. For example, if the data on the Windows system of origin is TS then when the user logins to the declassification system, the user must login with TS-level credentials in order to access a working environment capable of supporting the required level of classification. In addition, the data must be transferred from the transfer media to the declassification system at the same level of classification as the data itself. Under MLS systems, unlabelled data on external media devices will automatically assume the security or classification label and level of the environment in which the user mounts the device. For example, if the transfer data is unlabelled but is considered TS based on its classification on the originating system, then once the user has been granted appropriate rights and privileges to access the external transfer media, the user mounts the transfer media in a work environment that is at the same level of classification as the transfer data. Only then is the transfer data copied into the user's working directory where it will only be accessible by that user at that level of classification. No user-initiated processes at other levels of classification on the declassification system will be able to access the transferred data in the working directory, nor will lower-level work sessions be able to access the transferred data.

However, there is one issue with the aforementioned process. The problem is that the system or security administrator must be with the user the entire time in order to ensure that the proper system-specific procedures are followed for copying and maintaining the appropriate level of classification on the transfer data. In addition, the system or security administrator must be present to only grant and then revoke the various rights and privileges necessary to carry out the aforementioned actions. Therefore, this can be a time consuming process, especially if such requests are frequent. Once the data has been copied over, assuming with the proper data transfer protocols, the data can then be considered securely transferred. In addition, once the granted rights and privileges have been revoked, the user can no longer transfer the data; instead, it will remain in at its current classification level regardless of attempts to circumvent this by the user.

Data declassification requires the removal of all records, images and data that could potentially pose a security risk at the level of requested declassification. Although the user may make the necessary changes to the documents and other data files as he sees fit to accomplish the task, the system or security administrator should nevertheless have the right to review the actions undertaken. Furthermore, the local ISSO will surely examine these files before approving any final declassification as the final responsibility for declassification resides with the ISSO and the latter will decide the readiness of the requested files and data for declassification and dissemination. Although not all government departments have ISSOs, they certainly have an equivalent of some sort. In addition, the lower the level of requested declassification on a given set of files and data, the more work will be required to properly sanitize them, both on the part of the user and on the part of certifying authority who must thoroughly review the sanitized files and data.

Files and data marked for public release will obviously undergo the heaviest editing and purging due to the necessity to remove all potentially implicating information and details concerning classified activity. All data declassification should be conducted by the user while in the presence of a witness who can objectively attest that the necessary records, fields, tables, data, images, etc. have been expunged. However, this may or may not be a departmental policy.

Only after final approval has been granted from the local ISSO, departmental equivalent, or local management (depending on departmental regulations and policies) are the edited files marked for declassification and downgrading to the appropriate requested level. In order for data to be downgraded, the system or security administrator must assign specific rights and privileges to the user in order to gain the ability to downgrade files. Once granted and while in the presence of a witness and/or the ISSO, the appropriate files are downgraded to the corresponding requested level of classification. Once the process is completed, the system or security administrator must then revoke the user's newly assigned privileges.

Once downgrading is completed, the downgraded files can be placed onto permanent media such as CD or DVD by the user. The actual process for placing the downgraded files and data to releasable media will vary according to the specific rights and privileges of the user and the type of media to be used for the dissemination of the declassified files and data. However, it is very likely that the most common form of releasable media will be in the form of optical disk including CD and DVD. The authors highly recommend this form of media for releasing and transferring downgrading data and files.

If the user conducting the sanitization and downgrading of data and file does not have the appropriate system privileges required to copy the data and files to removable media, system security should not be compromised permanently granting the user the authority to do so. Instead, the user (author of data or files) can copy the downgraded files to a public directory (e.g. */tmp*) from where the system or security administrator can then burn them to releasable media. Once the data has been successfully archived onto the releasable media, the downgraded files, both the copies residing in the user's working directory (including the originals at the higher level of classification and the downgraded files and data) and those in the public directory, must be securely wiped from the system. Then, according to site or departmental policy, with the approval of the local ISSO, the releasable media may be taken out of the classified room and distributed accordingly.

For additional system security throughout the entire process of declassification and dissemination, the declassification system's swap space can be disabled, but only if enough system memory is available. If not, then upon completing the declassification and dissemination of the releasable media, the swap space should be thoroughly sanitized to ensure that any remnants in swap are permanently erased.

Throughout this memorandum, TSOL 02/04[8] was used on a Sun Microsystem's SPARC-based workstation, specifically an Ultra-10 workstation. The workstation was not connected nor configured for network use to avoid any potential source of user-based cross-contamination.

[8] 02/04 denotes that the distribution was released February 2004.

The NSA has provided a highly useful and informative guideline on sanitizing Microsoft-based documents and PDF files [22]. In addition, Microsoft has also provided a useful guide [23]. Both these guides should be used in conjunction with each other to maximize the underlying security of the declassified files. Of course, it is important that all files undergo a final inspection before release. During this inspection, it is useful to examine the files using a hex editor or other capable text editor to search for specific strings that were present at the files' previous levels of classification. For example, searching for keywords or phrases that were removed from the originals will help to ensure that all sensitive materials have been appropriately expunged. Since most files will be Microsoft Office generated, they can be easily searched due to the reliance of modern Microsoft Office implementations use of XML, thereby simplifying text-based searches. PDF files can be, with the appropriate tools installed, and converted directly to text or to another useful format for searching out patterns and text.

Depending on site and/or departmental policies and regulations, releasable files and data may have to be published to PDF-based documents. Such documents offer the ability to store additional metadata that can be used for traceability purposes. In addition, these documents can be encrypted or can have invisible watermarks overlaid to each document page, thereby significantly aiding the tracking and dissemination of released files and data.

Sanitization of electronic documents can be very challenging. The aforementioned guides [22, 23] provide many useful hints and tips for purging unnecessary information from electronic documents. However, it is often helpful to delete all unnecessary images, tables, forms, etc., from within a document. It is important that all files marked for declassification can be edited on the designated declassification system, as Windows-based systems do not distinguish between files at different levels of classification. Furthermore, in order for the files to be edited, especially if they are Microsoft Office-based, they should be converted into an equivalent OpenOffice format before beginning the task of declassifying any document. The Sun declassification system used throughout this memorandum uses OpenOffice because Microsoft Office does not work under Solaris. Before reconverting the modified files back to an Office-based format for downgrading and eventual dissemination, the actual OpenOffice files can be directly examined using a text editor. By default, OpenOffice stores its file formats using a compressed XML-based encoding scheme thus permitting easy visual examination of data files [68].

2 Installing and configuring TSOL and selected FOSS components

2.1 A brief note

All the following software packages were downloaded from the Internet, scanned for viruses, malware, spyware, etc. They were then burned to CD and loaded onto the TSOL declassification system. Different roles were used to install the various software packages, depending on the specific procedures required to install the various packages. The two roles used were Admin and Root, both of which used the ADMIN_LOW workspace label. Package installation specifics are found in their corresponding section below.

For the CD drive to be accessible, the role had to allocate the CD device, which was then automatically mounted for the role. Upon completion of a specific software installation, depending on the role used for the installation, the CD device may have to be de-allocated and ejected prior to reusing it to install other software as an altogether different role.

2.2 Installing and configuring TSOL

The installation of TSOL will vary according to the specifics of the underlying architecture and platform. An Intel-based installation will not be the same as a SPARC-based installation. It is highly recommended that the system not be connected to any network until the installation is complete and functional in order to avoid any potential source of contamination. For this memorandum, TSOL 8 02/04 was installed onto the following system:

- Sun Ultra-10
  - 384 MB RAM
  - two hard disk drives (7.36 GB and 7.00 GB, respectively)
  - two CD drives (1 CD-R and 1 CD-RW)

After powering on the system, at the system prompt, insert the first TSOL 8 02/04 CD into the CD drive and then type the following:

```
> probe-scsi-all

> boot cdrom
```

The above commands will probe the system bus for all connected I/O hardware and then boot the system from a recognized user-specified I/O device (e.g. CD). The system will then load from the provided CD and within several minutes boot-up into graphical mode, at which point the installation process will begin requesting input from the operator. Various pieces of information are required for a successful TSOL installation. In order of precedence, here are the various requested questions followed by their corresponding answers. It is possible that the reader's answers will vary from those provided here:

- Network: No network
- Host name: declass
- Support: Canada – English / USA
- Software group: Entire Distribution
- Disk: Manual Layout

  The manual layout of a disk will depend on the number of disks used, their size, swap requirements, size of log files to keep and organizational policy. This section should be filled in according to the reader's specific requirements.

  Disk 1: swap = 512 MB

  / = 6860 MB

  Disk 2: swap = 512 MB

  / = 6500 MB
- Remote File System: Skip Mount Remote File System
- Reboot: Auto Reboot
- Enter root password: RicardoRocks[9]
- Enter root password a second time: RicardoRocks
- Power Savings: No Power Saving and No asking every boot

The system will now reboot, detect and configure all system devices. After reboot and device configuration, the installation is complete. The system is now installed as a standalone workstation that is not connected to the network. It is completely independent of all other systems and is self-contained, thereby suitable for the storage of highly classified material. Software and data files that are to be worked on for declassification and downgrading are to be transferred using one of the machine's two CD drives. Data that is to be declassified and downgraded is written by the TSOL system using the CD-RW drive.

After the first initial boot-up, the graphical login window is presented and must be logged in to using the following information:

- Login user: install
- User password: install

This will provide the operator with access to the system, including the necessary system administration tools. Then system users and administrative roles must be created. Perform the following actions to configure users and roles:

- Open a terminal (right click on the screen -> menu -> tools -> terminal)
- Type: *smc &*

[9] Obviously these passwords must later be changed using Sun Management Console.

Since this is the first use of Sun Management Console, its start-up can be very long, approximately five to ten minutes depending on system load and processor speed. Once the system has started, the user must log on using the install user and its associated password. The install user has sufficient privileges and authorizations to create users and roles. Once logged into Sun Management Console, perform the following actions:

- Go to Users section and create users as necessary

  User accounts should be created on a one by one basis. In addition, at least two accounts will be required to provide access system and role access to the operators. These users should be attributed the Right "Enable Login". Without this right, no user can login to the system. Not all users should have this ability.

- Go to Roles section and create three roles:
  - Admin (admin):
    - Rights: System administrator
  - Secadmin (secadmin):
    - Rights: Information Security, Rights Security, ALL
  - Primary Admin (primaryadmin):
    - Rights: Primary Administrator, ALL

The root role already exists with the necessary rights. Different passwords should be used for each user and role, including the root role. The roles should be delegated and divided between at least two individuals so that no single person has all system privileges. Through task delegation, enhanced system security can be achieved. In this way, no single user or role has all capabilities.

Roles should then be assigned to key users. For example, assuming that two operators will co-manage the system, each operator will have his or her own user-specific account. Each account is then assigned specific roles that are used to manage different aspects of the system.

Finally, all users and roles should be thoroughly tested. It is also important to make certain, that at a minimum, key users can log into the system after a reboot and enable system logins for everyone else. Once completed using Sun management Console, the install user must be disabled as by default, it has too many authorizations and privileges to be left in active use. The system can of course be enhanced and improved through subsequent refinement of the system's configuration. However, it is important to note that the default security settings are sufficient to allow the system to have passed its Common Criteria LSPP and CAPP EAL4+ certification and accreditation.

As required, software can be installed and configured. So too can specific services and applications. The following two sections will be of use for installing and configuring additional software to provide increased capabilities to the system.

2.3 Installation of FOSS packages

2.3.1 Sunfreeware packages

In order to provide additional capabilities including software compilation (e.g. C/C++), networking tools and daemons (including SSH) and CD authoring tools, various FOSS packages have to be installed. These tools will serve to create data CDs for transferring and declassifying information (see Annex A) and creating a secure deletion program (see Annex B). This FOSS software will expand the inherent capabilities of TSOL by adding additional capacities commonly found in FOSS operating systems including Linux and BSD.

Oftentimes, compiling FOSS software can be arduous. It is even more arduous when done under proprietary operating systems including Solaris and TSOL. While these systems generally support the compilations of many FOSS applications and libraries based on Sun library files, dependencies are often found missing. However, much of the work has already been done by Solaris enthusiasts who precompile software for many different versions of Solaris, available both for the SPARC and x86 platform. This precompiled software can be found at Sunfreeware.

The files listed below were downloaded and installed on the TSOL declassification system to provide the aforementioned additional functionality and to satisfy various library requirements. The packages were installed using TSOL's *pkgadd* command; they can consequently be deleted using the *pkgrm* command. It is expected that the reader have basic understanding of package management under Solaris. Under TSOL, software management is the same as under Solaris.

The Sunfreeware packages were downloaded from the Sunfreeware. Subsequently, they were all scanned for viruses, rootkits and other malware. Then, they were burned to CD for transferring to the TSOL declassification workstation for installation. On the TSOL workstation, the files were copied from the CD to */tmp* for decompression and then installation. Using the Admin role the files were copied and installed.

The packages downloaded from Sunfreeware are as follows:

a2ps-4.13b-sol8-sparc-local.gz
antiword-0.37-sol8-sparc-local.gz
autoconf-2.62-sol8-sparc-local.gz
automake-1.10.1-sol8-sparc-local.gz
cdrtools-2.01.01a35-sol8-sparc-opt.gz
expat-2.0.1-sol8-sparc-local.gz
gcc-3.4.6-sol8-sparc-local.gz
gdb-6.6-sol8-sparc-local.gz
iplog-2.2.3-sol8-sparc-local.gz
jpeg-6b-sol8-sparc-local.gz
libiconv-1.11-sol8-sparc-local.gz
libintl-3.4.0-sol8-sparc-local.gz
libpcap-0.9.8-sol8-sparc-local.gz
libpng-1.2.29-sol8-sparc-local.gz
links-2.1pre36-sol8-sparc-local.gz

lynx-2.8.5-sol8-sparc-local.gz
make-3.81-sol8-sparc-local.gz
mc-4.6.1-sol8-sparc-local.gz
mgdiff-1.0-sol8-sparc-local.gz
mkisofs-1.12b5-sol8-sparc-local.gz
nc-110-sol8-sparc-local.gz
ncurses-5.6-sol8-sparc-local.gz
nedit-5.5-sol8-sparc-local.gz
netpbm-10.27-sol8-sparc-local.gz
openssh-5.0p1-sol8-sparc-local.gz
openssl-0.9.8h-sol8-sparc-local.gz
pcre-7.7-sol8-sparc-local.gz
pdftk-1.12-sol8-sparc-local.gz
pine-4.64-sol8-sparc-local.gz
pkgconfig-0.20-sol8-sparc-local.gz
pngcrush-1.6.5-sol8-sparc-local.gz
psutils-1.17-sol8-sparc-local.gz
slang-2.1.3-sol8-sparc-local.gz
ssldump-0.9b3-sol8-sparc-local.gz
tcpdump-3.9.8-sol8-sparc-local.gz
tcpflow-0.21-sol8-sparc-local.gz
tcpreplay-2.3.5-sol8-sparc-local.gz
tcptraceroute-1.5beta7-sol8-sparc-local.gz
tiff-3.8.2-sol8-sparc-local.gz
tree-1.5.1.1-sol8-sparc-local.gz

2.3.2 Java installation

This section is optional as OpenOffice includes its own bundled version of Java 1.6. However, its version is 32-bit only. Therefore, if 64-bit Java functionality is required, then the following instructions should be followed.

By default, TSOL comes preinstalled with Java versions 1.2 and 1.3, both of which are sufficient for many applications. However, other instances may necessitate a more recent version. Currently, the most stable release available for Solaris (and TSOL) is Java 1.6 Update 6. It is available for 32-bit or 64-bit systems. The 64-bit version, installed if 64-bit support is required, requires that the 32-bit version be installed first. However, the 32-bit version can be installed as is without 64-bit support.

The 32-bit Java 1.6 Update 6 file is named *jre-6u6-solaris-sparc.sh* and the 64-bit Java package is named *jre-6u6-solaris-sparcv9.sh*. Both files are shell archives and as such can be executed as though they were ordinary shell scripts. Both files are installed from the command line, as they do not come bundled with a graphical installer.

The files are copied from the download workstation to the TSOL system using CD. From the CD, they should be copied to */tmp*. Once copied, they should be executed from there. However, depending on whether 32 or 64-bit support is required, one or both files may be copied.

The packages must be executed and installed as the Root role. Attempting installation from any other role will not work. The 32-bit package must be run first, followed by the 64-bit package only if 64-bit support is necessary. Once both shell scripts are finished executing, they create Java software directory *jre1.6.0_06* found under */tmp*. Directory *jre1.6.0_06* should then be moved to */usr*. Once completed, user and profile environmental settings (e.g. *.profile*) should be reconfigured to point to the new Java installation.

2.3.3 OpenOffice installation

The OpenOffice packages are installed using the Admin role. For this specific installation, OpenOffice 2.4.0 was installed. The OpenOffice package is downloaded as a compressed archive and must be placed onto the TSOL workstation using a CD. Once the compressed OpenOffice package is copied from the CD to the system's */tmp* directory, it must be decompressed and its files extracted using *gzip* and *tar*, respectively. Extraction will generate a top-level directory under */tmp*. The current version of OpenOffice creates top-level directory */tmp/OOH680_m12_native_packed-1_en-US.9286*. Under this directory, several new directories and files are found. It is there, directly under the aforementioned top-level directory, that the various OpenOffice packages must be installed using the *pkgadd* command.

This version of OpenOffice comes bundled with a 32-bit version of Java 1.6. There are 29 packages in all to install and installation may take upwards of an hour, depending on the speed of the underlying system. Once installed, the user will have to log out and log back into the system in order to have direct and immediate access to OpenOffice, which is available directly from the CDE desktop task bar.

Once the various OpenOffice components are installed, it will then be possible to edit all Microsoft-based or compatible data files and electronic documents using OpenOffice. Furthermore, it will then be possible to redistribute those files using the same original file format or an altogether new one such as PDF (OpenOffice can publish directly to PDF).

2.4 User and role environment variables

This section is optional. However, after installing the various Sunfreeware packages, many additional libraries and executables have been added to the system. OpenOffice will work regardless of whether its specific environment variables are defined for a specific user or profile as it pre-set to run directly from the CDE taskbar. Of course, running OpenOffice directly from the command line will require setting the appropriate environment variables.

Java 1.6 Update 6 may be optionally installed and configured, depending on whether 64-bit Java support is required. If not, the 32-bit version of Java installed with OpenOffice may be used in lieu. The OpenOffice installation program makes OpenOffice directly available from the CDE taskbar.

For the Trusted Solaris installation outlined herein, no environmental variables were set for either OpenOffice or Java 1.6 as there was no specific requirement to run Java 1.6-based applications other than OpenOffice, which itself is run directly from the CDE taskbar. OpenOffice, by default, is installed and is automatically made available to all users and profiles on the declassification system.

Every user and profile can have his environmental variables set to configure a more usable workspace. User and profile environment variables are generally set using the file *.profile* located in each user or profile home directory. However, CDE environmental variables can be overridden by setting a user or profile's *.dtprofile* file. By default, however, CDE picks up its settings from *.profile*. Configuring either file is optional. It will depend more specifically on the needs of a given user or profile. For consistency, it is suggested that only *.profile* be modified to contain all necessary environmental variables and settings.

At a minimum, after installing all of the aforementioned software, a suggested *.profile* should contain these following entries:

```
PATH=$PATH:/etc:/usr/local/bin:/usr/local/sbin:/usr/local/libexec/gcc/sparc-sun-
solaris2.8/3.4.6:/usr/local/netpbm/bin:/usr/local/ssl/bin:/opt/schily/bin/opt/schily/sbin:/usr
/ccs/bin:/usr/ccs/bin/sparcv9

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:/usr/local/netpbm/lib:/usr/
local/ssl/lib:/opt/schily/lib

MANPATH=$MANPATH:/usr/local/man:/usr/local/netpbm/man:/usr/local/share/man:/u
sr/local/ssl/man:/opt/schily/man

HISTSIZE=1000000

HISTFILESIZE=1000000

export  PATH  LD_LIBRARY_PATH  MANPATH  HISTSIZE  HISTFILESIZE
```

By default, under both TSOL and Solaris, environment variables $PATH, $LD_LIBRARY_PATH and $MANPATH already exist and are minimally set with only the strictly necessary paths. The system administrator must ensure that each user and profile has immediate access to all commonly used programs and libraries. The aforementioned environmental variables and settings are a template and the reader is free to modify them as required or altogether omit.

3 Conclusion

Although many different computer operating systems can be used in classified environments, few are able to securely control and limit access to files and data. MLS-based systems accomplish this through an intricate but effective set of security mechanisms, access policies, rights and privileges, all of which work in concert to dramatically improve the overall security position of the system. Thus, in comparison to other computer operating systems, Trusted Solaris 8 provides a very rich set of security features enabling system and security administrators to tightly control access to classified files and data, including the downgrading and release of these files and data. User capabilities are entirely dependent on the rights and privileges assigned to them by the system or security administrator. In this way, a user never has more capabilities than necessary to carry out his assigned tasks and duties. It is for this reason that Trusted Solaris 8 is highly suitable for use in classified environments.

Trusted Solaris systems are far more secure than their unsecured counterparts including Solaris 10 and Microsoft Windows. This is because Mandatory Access Controls are built directly into the Trusted Solaris kernel to enforce compartmentalization and segregation at the lowest levels of the operating system. Therefore, in the opinion of the author, even while using unsecured software such as OpenOffice, these applications cannot, so long as default system security policies remain in place, leak classified information to other devices, users, or the network.

Although Solaris 10 now features Trusted Extensions, it has never been TCSEC certified and as such, the author is hesitant to recommend it for the uses outlined herein. Microsoft Windows-based systems are completely inappropriate systems to use for implementing a declassification-like solution similar to what has been examined herein. However, no direct comparison is currently available examining the technological and security differences between Trusted Solaris and Microsoft Windows.

Microsoft-based operating systems can achieve “pseudo” trusted capability only after investing large sums of money in third-party software (consider Green Hills Software) while Trusted Solaris remains relatively secure in comparison, at least as long as Trusted Solaris is not run on large multiprocessor datacentre-class computer systems as Sun Microsystems can change exorbitant amounts of money it.

Under Trusted Solaris, access to system resources including disk, memory, removable devices, printers and even the network can be fine-tuned or restricted, depending on the user. This is not to endorse Trusted Solaris as necessarily superior to other certified TOS systems.

Because of the inherent capabilities built into MLS MAC-based systems such as Trusted Solaris, these systems are inherently a good choice for use as declassification workstations. However, Sun RISC-based UNIX workstations cost considerably less than other brands. IBM systems are expensive relative to Sun’s budget line-up of workstations. Furthermore, SGI no longer markets their RISC MIPS-based UNIX systems any longer. Therefore, in the efforts of fairness and ease of access to various systems to work with, quality of system documentation and overall marketplace availability, Sun Trusted Solaris is, in the opinion of the author, a highly capable and secure operating system for carrying out classified work. Furthermore, due to the use of OpenOffice, which is available for Solaris and other platforms including Microsoft Windows, compatibility-based issues between UNIX and Windows systems are readily mitigated. OpenOffice enables users to retain compatibility with PC-based software systems, which today permeates government and military networks and installations alike.

Trusted Solaris enables system and security administrators to accurately define and implement their security policies as they pertain to the storage, treatment and redaction (e.g. sanitization) of classified data files so that they can be declassified or downgraded. This is made possible by Trusted Solaris' implementation of highly flexible and configurable security configurations, many of which can be edited using Sun's graphical management tool, Sun Management Console. Networking is highly configurable with respect to permissiveness and users files and data are by default limited to the confines of their own working directories at their respective levels of classification.

Unfortunately, the majority of classified systems are Windows-based, which are simply not up to the task. Even if these Windows systems are retrofitted with high-cost third-party software (e.g. Green Hills Software's Integrity PC), the system remains inherently insecure because MAC-based access policies are not implemented at the kernel level, even if this software adds kernel level extensions. The Windows kernel is not intrinsically up to the task nor is the rest of the operating system's components. These systems can therefore not be trusted for declassification work as there is no low-level access restriction and enforcement capability in place to hinder the comingling of highly classified data with data of lesser sensitivity labels. If such systems are used for declassification purposes, then there are no robust means of assuring that system processes, users or the system itself will not access restricted data and information when attempting to declassify specific data files and cause cross-contamination. Although to some this may seem like a stretch of reality, the fact remains that even with additional third-party software; little effectual mechanism exists to enforce a no cross-contamination policy of data and information under Windows because of the kernel's lack of ability to compartmentalize and segregate data and information. Additional software may change portions of Windows behaviour, but not sufficiently or to a level low enough to satisfy security conscientious IT professionals.

Therefore, in the opinion of the author, Trusted Solaris is suitable for use as a declassification workstation. Under such a context, classified data files are extracted from classified Windows-based workstations or networks and then edited, redacted and sanitized. Then, with appropriate approval, using a TSOL-based declassification workstation, the files are declassified and released.

The objective of this memorandum was to determine to what extent, if any, classified files could be declassified using a TOS system such as Trusted Solaris. As it turns out, using an office editing suite such as OpenOffice while under the confines of a MAC-enabled MLS system such as Trusted Solaris, files could be securely edited, redacted and sanitized, as long as certain guidelines are followed. Upon successful data file editing, files could be downgraded, with ISSO or managerial approval, to a lower level of classification for increased dissemination with a sense of confidence that only approved and redacted materials have been released.

References

[1] Wikipedia. Entry: Multi Level Security. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Multilevel_security.

[2] Runge, Chris. The Path to Multi-Level Security in Red Hat Enterprise Linux. Technical paper. Red Hat Inc. Document No: WHP174452UN. February 2006. http://www.redhat.com/f/pdf/sec/path_to_mlsec.pdf.

[3] Garlick, Timothy. Trusted Solaris System Administration Student Guide. Book. Revision D. Sun Microsystems Inc. Document No. SC-235. 2003.

[4] Wikipedia. Entry: Bell-LaPadula Model. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Bell-LaPadula_model.

[5] Sun Microsystems Inc. Trusted Solaris Operating System. Online resource – homepage. Sun Microsystems Inc. 2008. http://www.sun.com/software/solaris/trustedsolaris/index.xml.

[6] Hewlett-Packard Inc. Administering Your HP-UX Trusted System: HP 9000 Computer Systems. Book. First edition. Hewlett-Packard. HP Part No. B2355-90121. August 1996. http://docs.hp.com/en/B2355-90121/B2355-90121.pdf.

[7] Silicon Graphics Inc. Trusted Computing—The SGI® Solution. White paper. Silicon Graphics Inc. Document No: 3185. December 2001. http://www.sgi.com/pdfs/3185.pdf.

[8] IBM Inc. Trusted AIX. IBM online information center. http://publib.boulder.ibm.com/infocenter/systems/index.jsp?topic=/com.ibm.aix.security/doc/security/trusted_aix.htm

[9] McCarthy, Bill. SELinux: NSA's Open Source Security Enhanced Linux. Book. First edition. O'Reilly Media Inc. ISBN No. 0596007167. 2005.

[10] The FreeBSD Documentation Project. FreeBSD Handbook. Book. FreeBSD 7.0-RELEASE. The FreeBSD Documentation Project. 2008. http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/index.html.

[11] Faden, Glenn. Comparing the Multilevel Security Policies of the Solaris Trusted Extensions and Red Hat Enterprise Linux Systems. Online article. Sun Microsystems Inc. February 2007. http://www.sun.com/bigadmin/features/hub_articles/mls_trusted_exts.jsp.

[12] X.Org Foundation. X.Org Wiki – ProPolice. Online resource. X.Org Foundation. May 2008. http://xorg.freedesktop.org/wiki/ProPolice.

[13] Wheeler, David A. Secure Programming for Linux and Unix HOWTO. Online book. Version 3.010. David A. Wheeler. March 2003. http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/index.html.

[14] Wikipedia. Entry: Auditing. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Computer_security_audit.

[15] Wikipedia. Entry: Trusted Path. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Trusted_path.

[16] Wikipedia. Entry: Mandatory Access Control. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Mandatory_access_control.

[17] Wikipedia. Entry: Role Based Access Control. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Role-Based_Access_Control.

[18] Wikipedia. Entry: Access Control List. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Access_control_lists.

[19] Wikipedia. Entry: Discretionary Access Control. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Discretionary_Access_Control.

[20] Wikipedia. Entry: Principle of Least Privilege. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Principle_of_least_privilege.

[21] Silicon Graphics Inc. Multilevel Security (MLS) by Trusted IRIX™. White paper. Silicon Graphics Inc. Document No: 3241. November 2002. http://www.sgi.com/pdfs/3241.pdf.

[22] National Security Agency. Redacting with Confidence: How to Safely Publish Sanitized Reports Converted From Word to PDF. Technical paper. National Security Agency: Architectures and Applications Division of the Systems and Network Attack Center, Information Assurance Directorate. Document No: I333-015R-2005. February 2006. http://www.nsa.gov/snac/vtechrep/I333-TR-015R-2005.PDF.

[23] Microsoft. Office 2003 Add-in: Word Redaction v1.2. Online resource. Version 1.2. Microsoft. May 2006. http://www.microsoft.com/downloads/details.aspx?FamilyID=028c0fd7-67c2-4b51-8e87-65cc9f30f2ed&DisplayLang=en.

[24] Wikipedia. Entry: Sanitization. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Sanitization_%28classified_information%29.

[25] Wikipedia. Entry: Trusted Computer System Evaluation Criteria. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Trusted_Computer_System_Evaluation_Criteria.

[26] Michaud, F., Carbone, R. Practical Verification & Safeguard Tools for C/C++. Technical Report. Defence R&D Canada - Valcartier. Document # TR 2006-735. November 2007. http://cradpdf.drdc.gc.ca/PDFS/unc69/p528977.pdf.

[27] Painchaud, F., Carbone, R. Java Software Verification Tools: Evaluation and Recommended Methodology. Technical Memorandum. Defence R&D Canada - Valcartier. Document # TM 2005-226. February 2007. http://cradpdf.drdc.gc.ca/PDFS/unc57/p527369.pdf.

[28] Common Criteria. Common Criteria for Information Technology Security Evaluation: Part 1: Introduction and general model. Book. Version 3.1, Revision 1. Document No: CCMB-2006-09-001. September 2006. http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R1.pdf.

[29] Neugent, H. William, Olson, Ingrid M. and Alfred W. Arsenault. COMPUTER SECURITY REQUIREMENTS: GUIDANCE FOR APPLYING THE DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA IN SPECIFIC ENVIRONMENTS. (Light Yellow Book). DoD specification. DoD Computer Security Center. Document No: CSC-STD-003-85, Library No. S-226,727. June 1985. Light yellow book http://www.fas.org/irp/nsa/rainbow/std003.htm.

[30] Office of Standards and Products, National Computer Security Center. DEPARTMENT OF DEFENSE STANDARD: DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA. (Orange Book). DoD specification. Office of Standards and Products, National Computer Security Center. Document No: DoD 5200.28-STD, Library No. S225,711. December 1985. http://www.fas.org/irp/nsa/rainbow/std001.htm.

[31] National Computer Security Center. A Guide to Writing the Security Features User's Guide for Trusted Systems. (Hot Peach Book). DoD specification. Version 1. National Computer Security Center. Document No: NCSC-TG-026, Library No. 5-237. September 1991. http://www.fas.org/irp/nsa/rainbow/tg026.htm.

[32] Anderson, James, Vaughn, Rayford (Lt. Col. - USA). A Guide to Understanding Identification and Authentication in Trusted Systems. (Light Blue Book). Version 1. National Computer Security Center. Document No: NCSC-TG-017, Library No. 5-235,479. September 1991. http://www.fas.org/irp/nsa/rainbow/tg017.htm.

[33] Info Systems Technology Inc., Gligor, Virgil D. (Dr.). GUIDE TO UNDERSTANDING TRUSTED FACILITY MANAGEMENT. (Brown Book). DoD specification. National Computer Security Center. Document No: NCSC-TG-O15, Library No. S-231,429. June 1989. http://www.fas.org/irp/nsa/rainbow/tg015.htm.

[34] Zurschmeide, Jeffrey B.. Trusted IRIX/CMW Security Administration Guide (IRIX 6.5). Book. Revision 009. Silicon Graphics Inc. Document No: 007-3299-009. December 2003. http://techpubs.sgi.com/library/tpl/cgi-bin/browse.cgi?coll=0650&db=bks&cmd=toc&pth=/SGI_Admin/TCMW_AG.

[35] Zurschmeide, Jeffrey B., Johnson, Karen and Terry Schultz. Trusted IRIX™/CMW Security Features User's Guide. Book. Revision 005. Document No: 007-3300-005. December 2003. http://techpubs.sgi.com/library/tpl/cgi-bin/browse.cgi?coll=0650&db=bks&cmd=toc&pth=/SGI_EndUser/TCMW_UG.

[36] Hewlett-Packard Inc. Security-enhanced Linux: Information assurance with HP Integrity Servers and Red Hat Enterprise Linux. White paper. Hewlett-Packard Inc. 2006. http://h20219.www2.hp.com/enterprise/downloads/RedHatLinux_Integrity_wpSVK.pdf.

[37] Wikipedia. Entry: CTCPEC. Online encyclopaedic entry. Wikimedia Inc. May 2009. http://en.wikipedia.org/wiki/CTCPEC.

[38] Sun Microsystems Inc. Trusted Solaris User's Guide. Book. Sun Microsystems Inc. Part No: 805-8115-10. http://docs.sun.com/app/docs/doc/805-8115-10.

[39] Sun Microsystems Inc. Trusted Solaris Label Administration. Book. Sun Microsystems Inc. Part No: 805-8122-10. http://docs.sun.com/app/docs/doc/805-8122-10.

[40] Sun Microsystems Inc. Trusted Solaris Installation and Configuration. Book. Sun Microsystems Inc. Part No: 805-8114. http://docs.sun.com/app/docs/doc/805-8114.

[41] Sun Microsystems Inc. Trusted Solaris Administrator's Procedures. Book. Sun Microsystems Inc. Part No: 805-8120-10. http://docs.sun.com/app/docs/doc/805-8120-10.

[42] Sun Microsystems Inc. Trusted Solaris Administration Overview. Book. Sun Microsystems Inc. Part No: 805-8119-10. http://docs.sun.com/app/docs/doc/805-8119-10.

[43] Klinker, J. Eric, Mihelcic, David M. REDUCING THE RISK OF MULTI-LEVEL SECURE (MLS) WORKSTATIONS. Technical paper. Naval Research Lab. ISBN No: 0-7803-4249-6. Presented to IEEE 1997. http://ieeexplore.ieee.org/iel3/5148/13984/00645002.pdf?arnumber=645002.

[44] Common Criteria. Certified Product List. Evaluated product list. Common Criteria Portal. July 2008. http://209.85.165.104/search?q=cache:1J92GZMzeTkJ:www.commoncriteriaportal.org/products_OS.html+common+criteria+certified+products+operating+systems&hl=en&ct=clnk&cd=2&gl=ca (Cached Google list).

[45] Australian Government Department of Defence. Historical Evaluated Products List. Product evaluation list. Australian Government Department of Defence. 2007. http://www.dsd.gov.au/infosec/evaluation_services/epl/historical.html.

[46] Wikipedia. Entry: Common Criteria. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Common_Criteria.

[47] Sun Microsystems Inc. Trusted Solaris™ 8 Operating Environment: A Technical Overview. Technical paper. Sun Microsystems Inc. November 2000. http://www.sun.com/software/whitepapers/wp-ts8/ts8-wp.pdf.

[48] Sun Microsystems Inc. Solaris Trusted Extensions Administrator's Procedures: Trusted Networking (Overview). Book. Sun Microsystems Inc. Part No: 819-7309-05. May 2008. http://dlc.sun.com/osol/docs/content/TRSOLADMPROC/txnet-1.html.

[49] Wikipedia. Entry: Trusted BSD. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/FreeBSD#TrustedBSD.

[50] Wikipedia. Entry: SELinux. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/SELinux.

[51] Vance, Chris, Watson, Robert. Security Enhanced BSD. Technical paper. Network Associates Laboratories. Work supported by DARPA/SPAWAR contract N66001-01-C-8. July 2003. http://www.trustedbsd.org/sebsd-july2003.pdf.

[52] Brindle, Joshua, Vance, Karen and Chad Sellers. Enforcing Security Enhanced Linux policies in a networked policy domain. Technical paper. Tresys Technology. 2007. http://selinux-symposium.org/2007/papers/02-pms.pdf.

[53] Morris, James. Networking in NSA Security-Enhanced Linux. Technical paper. Red Hat Inc. Originally published in The Linux Journal. January 2005. http://www.redhat.com/f/pdf/sec/Networking_in_NSA-SELinux.pdf.

[54] Wikipedia. Entry: Trusted Operating System. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Trusted_operating_system.

[55] Wikipedia. Entry: Type Enforcement. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Type_enforcement.

[56] Watson, Robert N. M., Salamon, Wayne. The FreeBSD Audit System. Technical paper. TrustedBSD project. 2006. http://www.trustedbsd.org/20060303-ukuug2006lisa-audit.pdf.

[57] Watson, Robert, Vance, Chris, et al. The TrustedBSD MAC Framework: Extensible Kernel Access Control for FreeBSD 5.0. Technical paper. Network Associates Laboratories. Originally presented to USENIX 2003. 2003. http://www.trustedbsd.org/trustedbsd-usenix2003freenix.pdf.

[58] Habib, Irfan. Creating SELinux Policies Simplified. Online column. The Linux Journal. February 2007. http://www.linuxjournal.com/article/8766.

[59] Smalley, Stephen, Vance, Chris and Wayne Salamon. Implementing SELinux as a Linux Security Module. Network Associates Laboratories and National Security Agency. http://www.nsa.gov/SELinux/papers/module/t1.html.

[60] National Security Agency. Configuring the SELinux Policy. Technical paper. National Security Agency. http://www.nsa.gov/selinuX/papers/policy2/t1.html.

[61] Red Hat Inc. Deployment Guide 5.0: Red Hat Enterprise Linux. Book. Red Hat Enterprise Linux 5.0.0. Red Hat Inc. January 2008. http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/en-US/Deployment_Guide.pdf.

[62] Carbone, Richard, Charpentier, Robert and David Demers. Free and open Source Software (FOSS) in Military Computing. Technical paper. Defence R&D Canada. Prepared for 10th International Command and Control Research and Technology Symposium. Track Topic: C4ISR/C2 Architecture. June 2005. http://www.dodccrp.org/events/10th_ICCRTS/CD/papers/227.pdf.

[63] Charpentier, Robert, Carbone, Richard. Free and Open Source Software: Overview and Preliminary Guidelines for the Government of Canada. External client report. Defence R&D Canada. DRDC-Valcartier-ECR-2004-232. December 2004. http://www.tbs-sct.gc.ca/fap-paf/oss-ll/foss-llo/foss-llo00-eng.asp.

[64] Department of Justice Canada. Criminal Code Act. Canadian Parliamentary Law. Act C-46. Department of Justice Canada. http://laws.justice.gc.ca/en/C-46.

[65] Wikipedia. Entry: Declassification. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Declassification.

[66] Wikipedia. Entry: Freedom of Information Legislation. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Freedom_of_information_legislation#Canada.

[67] Department of Justice Canada. Access to Information Act. Canadian Parliamentary Law. Act A-1. Department of Justice Canada. http://laws.justice.gc.ca/en/ShowFullDoc/cs/A-1///en.

[68] Sun Microsystems Inc. Advantages of the OpenOffice.org XML File Format Used by the StarOffice™ Office Suite. White paper. Sun Microsystems Inc. April 2004. http://www.sun.com/software/whitepapers/staroffice/StarOfficeXML_wp042204.pdf.

[69] St. Johns, M. RFC 1038 - Draft revised IP security option. Request for Comment. RFC 1038. Network Working Group, IETF. January 1988. http://www.faqs.org/rfcs/rfc1038.html.

[70] Wikipedia. Entry: Transmission Control Protocol/Internet Protocol. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Internet_protocol_suite.

[71] Bunten, Andreas. UNIX and Linux based Rootkits Techniques and Countermeasures. Technical paper. DFN CERT. April 2004. http://www.first.org/conference/2004/papers/c17.pdf.

[72] Wikipedia. Entry: Buffer Overflow Protection. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Stack-smashing_protection.

[73] Common Criteria. Common Criteria for Information Technology Security Evaluation: Part 2: Security functional components. Book. Version 3.1, Revision 2. Document No: CCMB-2007-09-002. September 2007. http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R2.pdf.

[74] Common Criteria. Common Criteria for Information Technology Security Evaluation: Part 3: Security assurance components. Book. Version 3.1, Revision 2. Document No: CCMB-2007-09-003. September 2007. http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R2.pdf.

[75] Loscocco, Peter A., Smalley, Stephen D, Muckelbauer, Patrick A., et al. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. Technical paper. National Security Agency. http://www.nsa.gov/selinux/papers/inevitability/.

[76] Sun Microsystems Inc. Bank One Applies Military-Grade Security to Payment Processing. Online article. Sun Microsystems Inc. October 2003. http://www.sun.com/br/0104_ezine/fs_bankone.html.

This page intentionally left blank.

Annex A Downgrading and CD burning under TSOL

A.1 Downgrading files

Once the classified files have been copied from the source network or workstation using the aforementioned transfer media, they must be copied to the TSOL declassification workstation to the user's work directory under the same level of classification as the transfer data itself resides w. Using OpenOffice, the transferred files and data are edited and modified following the guidelines put in place by the NSA and Microsoft [22, 23]. These files can then be published to PDF or reconverted back to Microsoft Office format for release, depending on specific departmental procedure and policy. Optionally, once the OpenOffice document files have been sanitized, they can then be verified using a text editor. Each OpenOffice file is itself a series of compressed XML files that are easily examined. Once each OpenOffice file has been decompressed and its files extracted, they can be searched for patterns and strings or simply read as a document all the while looking for data that should have been purged. The editing of all files should be done while the user is accompanied by a witness in order to ascertain that the work was done thoroughly. However, this may be dependent on departmental procedure and policy. Furthermore, it is important that no declassification be attempted until authority has been granted either from the local ISSO or management as doing otherwise could constitute a breach of security protocol.

The transferred files, after having been opened and saved to OpenOffice format, will likely have to be reconverted back to their original format (generally Microsoft Office) or to a universal format, such as PDF, from OpenOffice before downgrading and release for dissemination can occur. The user does not have the rights and privileges necessary to accomplish either task. Specifically, only the declassification system's "secadmin" role (see Section 2.2) has the privilege to assign the right “Object Label Management” to a user. To do this, the user must have authorization to assume the secadmin role[10], otherwise a different user account (e.g. a system or security administrator user-based account that has the appropriate assumed authorization) will have to be used to login to the declassification system and then assume the secadmin role. Using Sun Management Console (system command *smc*), secadmin can temporarily assign this privilege to the user in question. Once the appropriate right has been granted to the user by secadmin, depending on whether the user had the right to assume the secadmin role or whether a system or security administrator user account had to be used, the user may have to log back in to the declassification system. Then, using *File Manager*, the user can drag and drop files from a higher-level security context or classification to a lower level one. These actions should only be done when accompanied by a witness and the express consent of the ISSO (or management) has been given. Furthermore, the ISSO may wish to be present when files are downgraded.

Once the appropriate files have been downgraded, the secadmin role must be reassumed and using Sun Management Console, (system command *smc*) the right “Object Label Management” must be removed from the user. The user must then log out and log back in order for these changes to take effect. The original and higher classified files must then be securely erased from disk (see Annex B).

[10] Generally, it is not a good idea to allow a user to ever assume a system or security administration role such as "primaryadmin" or "secadmin." However, departmental procedure and policy may allow for this.

It is important to remember that under TSOL, a user can be logged in to CDE and in a current session and have different applications such as *File Manager* running at each user-associated security context (classification level). Under TSOL, different security contexts or classifications are possible within any given CDE desktop window. This makes it easy, once the user has the right to do so, to transfer files from a higher classified context to a lower one.

The files are then ready for transfer to the removable media (e.g. burning to CD or DVD).

A.2 Settings permissions on *cdrecord*

Before *cdrecord* can be used by the root user to burn data to removable storage including CD or DVD, specific privileges must be assigned to the *cdrecord* executable. These privileges must be explicitly set as follows:

file_dac_read

file_mac_read

file_mac_write

sys_config

sys_devices

These permissions can only be set by the secadmin, primaryadmin or root role. The admin role does not have, by default, the system privileges to assign privileges to system executables. The executable *cdrecord* is found under */opt/schily/bin*.

A.3 Burning a CD under TSOL

Data CDs can be created under TSOL that are compatible with many other operating systems (e.g. Windows XP and Vista), including CD and DVD drives available from a variety of manufacturers. CD and DVD creation is made possible through the Sunfreeware software installed in Section 2.3.1. Although the process for DVD creation is similar, it is not explicitly explored in this section. Instead, the reader can familiarise himself with at his leisure.

CD creation requires that the copied data files and electronic documents have already been edited, verified and downgraded to the appropriate level by the user. Once this has been done, a CD image of the files to be transferred for declassification can be created. It is important that once the files have been downgraded to the appropriate level, the originals be securely deleted. Then the user should create a directory under which all his downgraded data files and documents will be temporarily stored. This makes creation of the CD image easier. Only the user himself can perform these actions under his own home directory as no other system user or role has the ability to access the user's files.

It is important that all actions undertaken by the user be done at the specific level of classification under which the data files are to be released. For example, if the original files were labelled as

TS and after editing they are considered S, then upon following their modification and downgrading, all the corresponding files should be located under the user's S labelled home directory[11] [12] where the CD or DVD image is to be created.

Once the files have been downgraded, the user should create a container directory within his home directory at the appropriate level of classification where all the files for release and declassification will be temporarily stored for creating the image file. This step is not necessary, but does facilitate the process of creating the image file. The image file is created using the *mkisofs* command found under */opt/schily/bin*. An example follows:

```
# mkdir container

# cp data_files container

# secure_delete data_files

# mkisofs –R –o cdimage.raw container/
```

Only the root user can burn data to CD or DVD. Assigning privileges and permissions to users or any other role in order to access CD or DVD writing capability would altogether break system security. Therefore, once the image file is created, it must be copied to a public directory where the root user can access it, such as */tmp*. From there, the file can be burned to CD or DVD and then is securely deleted. However, the root user must be running under the same level of classification as the image file is labelled, as otherwise, it will not be accessible, even to the root role. This is because of Write Up – Read Down (WURD). It is therefore important that the image file be labelled as S and that the root role switches from the default ADMIN_LOW label to S, so that the image can be accessed. Then the files can be burned directly to the device. Consider the following example:

As the user (running under S label):

```
# cp cdimage.raw /tmp

# secure_delete cdimage.raw
```

As the root role, under CDE, switch from ADMIN_LOW label to a corresponding image file label (e.g. S label):

```
# cd /tmp
```

[11] It is important to remember that under an MLS system, various levels of classification exist simultaneously. Depending on a user's current level of classification, which can always be changed between the various levels afforded to that user, only files of that classification can be accessed. Of course, a user can have multiple levels of classification running under a current session, as it is possible with TSOL. This makes possible dragging and dropping files when the corresponding system privileges have been accorded to the user, so that files from a different classification can be downgraded or upgraded.
[12] Different TOS systems behave differently. TSOL allows a user to run not only under multiple classifications, but also to concurrently have multiple sessions running at different classification levels. Other systems vary in their ability to provide this capability.

```
# cdrecord –v speed=2 dev=3,0 cdimage.raw
```

The speed will depend on the capability of the CD or DVD drive. The device number will vary according to its unique bus identifier. To determine the drive device number, the following command can be used:

```
# cdrecord –scanbus
```

At this time, the image file should have begun the transfer to the CD or DVD device. Once completed, the image file must be securely deleted. The disc is then ejected, labelled as S (or other appropriate security classification) and then transferred to the appropriate system, assuming it has the same or equivalent security classification.

It is important that all steps taken in this section be conducted in the presence of a witness and under the authority of the ISSO. Of course, specific sites may have their own rules and policies for carrying out this work. In either case, it is important that ISSO be aware of all activities concerning data declassification and be involved from the beginning, as the ISSO is the officer responsible for information security. Furthermore, data cannot generally be taken out from a highly classified environment into a less secure environment without the explicit consent of the ISSO. Of course, different sites may have their own policies in place for handling this situation.

Annex B Securely deleting files under TSOL

The importance of securely deleting data cannot be overstated. If, for any reason, the hard disk of the declassification system is stolen or lost, any highly classified material that were to be found could have devastating effects. In addition, modern computer forensic software and hardware are affordable and common enough today that anyone knowledgeable of the field could recover and repair data files from a misappropriated disk. Simply because the command *rm* reports a file as deleted does not actually indicate that it is in fact erased. Instead, the inodes that point to the file's location on disk are deleted. The data itself, although not directly accessible by the system, is still available to anyone capable of using advanced file recovery software. Therefore, it is vitally important that all files, when no longer needed, be permanently erased.

Under Windows, this is not an issue as there are literally hundreds of programs that are capable of securely and permanently erasing data from a drive. Under UNIX, it is less obvious. However, a highly recommended tool is THC Secure Delete 3.1 available from THC's web site. Once downloaded and transferred to the TSOL declassification system, the archive must be decompressed, and then the archive is extracted, compiled and installed. The user must assume the root profile to compile and install the tool. The tool's file name is *secure_delete-3.1.tar.gz* and once placed onto CD for transferring to the TSOL workstation, it should be copied to */tmp*.

Once the files have been uncompressed and extracted to directory */tmp/secure_delete-3.1*, the package can be compiled using the *make* command. Tests have revealed, however, that compilation succeeds for most but not all of the programs found in the secure deletion toolkit. This is, however, of no consequence, as the required tools do in fact successfully compile. The compilation errors are due to missing library files, which do not natively exist for Solaris, but only for Linux. Nevertheless, this small issue does not need to be immediately resolved in order to gain the desired functionality from the secure deletion toolkit. It is important, however, that all the packages from Section 2.3.1 be installed in order to successfully compile the majority of the tools within Secure Delete. The files that are to be copied to */usr/local/bin* are:

sfill

smem

srm

sswap

The accompanying man files will have to be copied over to */usr/local/man*. It will be necessary to alter the permissions of */usr/local/bin*, */usr/local/man* and */usr/local/man/man1* using "*chmod 777*" in order to have write permission to these directories. Once finished moving the files to their appropriate destinations, reset the permissions of the aforementioned directories using "*chmod 755*."

The tool for securely deleting files is *srm* and is likely to be the most commonly used command from the toolkit. By default, it carries out more than 30 passes on a given file in order to securely delete it. However, the larger the file, the longer this will take. Therefore, for the sake of expediency, less wipes may be used although this will depend on organizational or department data wiping procedures.

The program *sswap* can be used to securely wipe swap space. It requires that swap space be un-mounted prior to use and it will take a large amount of time to completely wipe the swap space. The larger the swap device, the more time it will take. Unfortunately, it is not an easy task predicting the amount of time required to adequately wipe swap space. The three most likely variables to affect the amount of required time for wiping swap space will be the size of the swap space, its location, if it is spread across multiple disks and the speed of the disk. Therefore, it is best to schedule swap space wipes when the system is not in use. For example, weekends and holidays are a good time to carry out these actions. The root role will be required to perform swap space wipes. However, the admin role will be required in order to disable and re-enable swap as well as to reformat the swap device. The only way to access a role is through CDE. Therefore, the system will have to remain at runlevel 3. It is for this reason that all users (except the currently logged in user) be logged off and all unnecessary applications and services be disabled. Once the swap space is wiped, the admin role will have to reformat the swap device using the *mkfs* command in order to recreate the logical structure destroyed by the wiping process.

Finally, the tool *sfill* can be used to wipe unused hard drive space (e.g. free space). This is a useful tool to use when it is suspected that users have not been using *srm* consistently or that application managed temporary files[13] were possibly used. Since these temporarily files are generally not securely deleted, it is possible that over time, especially if the system is heavily used or very large files are edited and modified, that the file system becomes cluttered with deleted but non-wiped data. This data could be recovered if the disk were removed from the system and forensically analyzed. However, under normal circumstances, while the disk resides within the machine, low-level sector-based access to the file system is not allowed, unless system security is broken by over-delegating privileges and authorizations to a role. The command *sfill* securely wipes all unused disk space. To maximize unused disk space from miscellaneous temporary files created across the system as users and applications interact with one another, unused disk space wipes should only be carried out, if possible, when no users are logged into the system. The root role will be required to perform this action and depending on available disk space, a successful pass could take several days to complete; therefore, this should only be scheduled on weekends and holidays. In addition, as a final note, it is possible to use *sfill* as a replacement to *sswap* for erasing swap devices. It is important, however, that before using it for purging swap space, unnecessary files in swap are deleted and that no running processes, services, or applications create or modify files on swap throughout the duration of the purge.

[13] It is common, even under UNIX, for applications to both create and delete many temporary files while they carry out their assigned tasks.

Annex C Viruses, rootkits and updates

C.1 Viruses

Few open source virus scanners exist for Solaris. The most popular of those is ClamAV, which, if compiled correctly and all dependency requirements are met, will work as expected. It will not be examined in this memorandum, as this is far too complicated and long to carryout herein. Commercial anti-virus scanners also exist for Solaris and include McAfee, F-Secure, Avast and several others. Since each of these is obtainable commercially for a fee, neither their installation nor use is examined herein either. It is noteworthy to point out that after numerous searches, no single virus could be found that would affect or infect Trusted Solaris.

C.2 Rootkits

To date, no known rootkit specifically affects TSOL. However, it is more likely that data transferred from another workstation, particularly if it is Windows-based, may have been compromised by a rootkit. Due to the fact that rootkits rarely affect UNIX systems, rootkit scanning technology under UNIX is not as sophisticated as it is for Windows, since mostly Windows systems are afflicted by rootkits. One highly useful and freely available rootkit scanner is *chkrootkit*, which is available from Sunfreeware. It was not included in the list found in Section 2.3.1, since this section is optional. The latest package at time of this writing (circa 2008), *chkrootkit-0.45-sol8-sparc-local.gz* is currently available. Once transferred to the TSOL declassification workstation, it can be installed. The admin role must be used to install the package. However, scanning with the tool must be done using the root role and it can only scan the entire directory structure. It cannot be used to scan specific user specified directories. [71]

It is very unlikely that a rootkit will successfully exploit a TSOL-based system because so long as no privileged role accesses and executes an affected file, the rootkit will not attach itself into memory or to another running process. MLS systems using MAC are particularly well secured against rootkits. Nevertheless, all software suffers from programming flaws and any potentially well-designed UNIX-based rootkit could take advantage of a UNIX common flaw and affect even TSOL. It is unknown, however, how much damage a successful rootkit can cause compared to a standard UNIX-based rootkit, as root-based processes are limited in their scope. Unfortunately, there is no public data to base any conclusion on, other than stating that there is no known TSOL-based rootkit. Using a preventive measure such as *chkrootkit* nevertheless makes good sense. [71]

C.3 Updates

The version of TSOL used throughout this memorandum is version TSOL 8 02/04, which is the most up-to-date version currently available from Sun Microsystems. It includes the large majority of patches and updates made to TSOL 8 since its inception. Since it is still available from Sun, it continues to be a supported product and hence enjoys product security patches when available and if applicable. It is important to understand that many security patches for standard Solaris 8 do not typically apply to TSOL 8 and applying them could replace enhanced executables and libraries with unsecured versions. It is therefore important to read all patch documentation before attempting to patch any security issue on TSOL, as each specific TSOL patch requires specific installation procedures.

At the time of this writing (circa 2008), there are about 15 patches currently available for TSOL 8 02/04. About half are available to the public for download and installation, while the other half requires a valid support contract.

Sun has no plans to release a TSOL 10 to match their current offering of Solaris 10, the current non-secured Sun-based operating system. Instead, Sun has ported many of the various technological aspects of TSOL 8 to Solaris 10 and released them as additional for-pay add-ons. However, even though Solaris 10 is superior in ways to its predecessor operating systems, even with its "secure add-ons" it does not yet implement half of the various security-conscious technologies found in TSOL 8. For this reason, only TSOL 8 should be considered for high-security data compartmentalization and segregation[14].

[14] At the time of this original writing which in 2008, this was true. However, Sun Microsystems has since been taken over by Oracle and the more recent versions of Solaris do support the vast majority of capabilities which TSOL offered, albeit for an additional cost. However, Solaris is no longer evaluated according to TCSEC and is now evaluated against the Common Criteria specification only.

Bibliography

Wikipedia. Entry: Common Desktop Environment. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Common_Desktop_Environment.

Wikipedia. Entry: OpenOffice.org. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/OpenOffice.

Wikipedia. Entry: National Institute of Standards and Technology. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/National_Institute_of_Standards_and_Technology.

Wikipedia. Entry: Portable Document Format. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Portable_Document_Format.

Wikipedia. Entry: Runlevel. Online encyclopaedic entry. Wikimedia Inc. July 2008. http://en.wikipedia.org/wiki/Runlevel.

List of symbols/abbreviations/acronyms/initialisms

ACL	Access Control List
AIX	Advanced Interactive eXecutive
ATM	Asynchronous Transfer Mode
BSD	Berkeley Software Distribution
CAPP	Controlled Access Protection Profile
CC	Common Criteria
CD	Compact Disc
CD-R	Compact Disc – Read Only
CD-RW	Compact Disc – Read/Write
CDE	Common Desktop Environment
CIPSO	Commercial IP Security Option
CSE	Canadian Security Establishment
CTCPEC	Canadian Trusted Computer Product Evaluation Criteria
DAC	Discretionary Access Control
DND	Department of National Defence
DoD	Department of Defense
DRDC	Defence Research & Development Canada
DVD	Digital Video Disc
EAL	Evaluation Assurance Level
FOSS	Free and Open Source Software
FreeBSD	Free Berkeley Software Distribution
GB	Gigabyte
GCC	GNU C Compiler Collection
GUI	Graphical User Interface
GNU	GNU's Not UNIX
HP-UX	Hewlett-Packard-UNIX
I/O	Input/Output
IBM	International Business Machine
IEEE	Institute of Electrical and Electronics Engineers
ISSO	Information Systems Security Officer

IT	Information Technology
LSPP	Labelled Security Protection Profile
MAC	Mandatory Access Control
MB	Megabyte
MDN	Ministère de la Défense nationale
MLS	Multi-Level Security
MS	Microsoft
NIST	National Institute of Standards & Technology
NSA	National Security Agency
PDF	Portable Document Format
PROM	Programmable Read-Only Memory
R&D	Research & Development
RAM	Random Access Memory
RBAC	Role-Based Access Control
RIPSO	Revised Interconnection Protocol Security Option or Revised Internet Protocol Security Option
RISC	Reduced Instruction Set Computer
S	Secret
SEBSD	Security-Enhanced BSD
SELinux	Security-Enhanced Linux
SGI	Silicon Graphics Inc.
SPARC	Scalable Processor Architecture
SSH	Secure Shell
Su	Switch User or Substitute User
Sudo	Superuser Do
TCP/IP	Transmission Control Protocol/Internet Protocol
TCSEC	Trusted Computer System Evaluation Criteria
TE	Type Enforcement
TM	Technical Memorandum
TN	Technical Note
TOS	Trusted Operating System
TRIX	Trusted IRIX
TrustedBSD	Trusted Berkeley Software Distribution

TS	Top Secret
TSOL	Trusted Solaris
USB	Universal Serial Bus
W3C	World Wide Web Consortium
WURD	Write Up – Read Down
XML	eXtensible Mark-up Language

Glossary

Access Control List (ACL)

An Access Control List is a list of specific privileges users and systems can have. These privileges are generally based on system access, file access, network access or access to various services. ACLs exist in various forms. Under MLS systems, ACLs can include files, directories, processes and the network accesses.

Accounting

Also known as system accounting, this is the process whereby system logs of system activity are kept for future use. System accounting logs are often used for keeping track of user logins and basic activity records including I/O and process activity. These logs are generally kept in */var/log* although their location can be modified. Accounting logs are often used for determining system use and basic troubleshooting.

Advanced Interactive eXecutive (AIX)

AIX is IBM's proprietary UNIX operating system, based mainly on System V UNIX and portions of BSD. It is a highly secure and scalable UNIX operating system. It is available on high-end IBM servers, datacenters and some IBM workstations.

Auditing

Also known as system auditing, this is the process whereby changes made to the system, its state, files or processes by users and services are logged for future use and security analysis. System auditing can be finely tuned and encompass user actions to changes made to system states, files and processes. These logs are generally used for auditing security events including, but not limited to, anomalies, aberrant system behaviour and potential security intrusions.

Bell-LaPadula Model

A model developed in 1973 by David Elliott Bell and Len LaPadula that formalized for DoD the basic underpinning and functionality of an MLS-based system. This model also proposes the basic functionality of a MAC-based system used for implementing MLS functionality.

Berkeley Software Distribution (BSD)

Berkeley Software Distribution is a very popular flavour of UNIX that was first developed at the University of California at Berkeley. BSD is very well known for its robust TCP/IP stack, excellent stability, scalability and robustness. There are currently three popular open source versions of the BSD operating system: FreeBSD, NetBSD and OpenBSD.

Buffer overflow

This is also commonly known as a buffer overrun. A buffer overflow is generally an inadvertent programming error that is introduced and often caused by the inappropriate use of pointers. User-specified input data or code can trigger a buffer overflow by running past the pointer and eating up the pointer's buffer memory. It is at this point that specially crafted code can trigger remote code execution (e.g. remote administrative command shell). All modern pointer-specific programming languages (e.g. C, Assembler, etc.) use pointers to modify directly memory or I/O devices. Modern pointer-free languages (e.g. Java, C#, etc.) are rarely are affected by buffer overflows. Buffer overflows are commonly found in operating system services, network protocols, stacks and applications. Depending on the nature of the overflow and the specificity of the malicious code, execution of the code may result in a system being successfully compromised.

C

The C programming language was developed at AT&T Bell Labs in 1972 by Dennis Ritchie. It is a procedural programming language still very popular to this day. It is commonly used in operating system development and UNIX-like operating systems are written almost exclusively in it.

C++

C++ is a superset of extended features for the C programming language. Based on C and Simula, it provides an all-purpose programming language that is object-oriented in nature and capable of performing all the same tasks as C. However, it is not meant to be a procedural language, although it can be used as such. Within the programming language, objects are used instead of procedures, providing the bulk of the functionality.

Common Desktop Environment (CDE)

CDE is a standard UNIX graphical desktop complete with necessary functionality for a functioning GUI environment complete with mouse (or other similar device) pointing and clicking capabilities. It is standard on many UNIX environments including those from SUN, IBM and HP.

Commercial IP Security Option (CIPSO)

CIPSO is a TCP/IP-specific security label-based encapsulation protocol useful for the open exchange of classified information between different TOS systems. The label of the packet is based on the security label of the data targeted for transmission. See RIPSO.

Common Criteria (CC)

Common Criteria is a new replacement for the TCSEC standard that evaluates hardware and software systems that process/store sensitive, protected or classified information.

Communications Security Establishment Canada (CSEC)

CSEC is Canada's cryptographic, communications and computer security organization, performing its functions on behalf of the Canadian Department of National Defence and the Government of Canada.

Canadian Trusted Computer Product Evaluation Criteria (CTCPEC)

This is the Canadian equivalent of TCSEC although it actually is a combination of the American TCSEC standard and the European ITSEC standard. It is considered by some as a compromise between both standards, although it has since been superseded by the internationally accepted CC standard. CTCPEC was put forward for Canada by CSEC in 1993.

Device Allocation

Under MLS systems, access to specific devices such as floppy, CD and DVD drives, as well as network require specific authorizations. Without these authorizations, these devices cannot be used for storing data or transmitting information. Through one or more operating system-specific mechanisms, the system administrator can define the devices and the permissions or authorizations different users can have with those devices. Therefore, Device Allocation is another mechanism for ensuring strict access controls.

Discretionary Access Control (DAC)

Contrary to MAC, DAC is a form of file permission and access control that normally allows users to control who accesses their files and resources. Users are able to modify permissions and access controls in order to fine-tune them to their needs.

Department of Defense (DoD)

DoD is the American defence department.

Department of National Defence (DND)

DND is the Canadian defence department.

Evaluation Assurance Level (EAL)

EAL is a common notion for describing the level of testing and verification a computer network operating system or networking hardware device has undergone to determine its overall security profile. Testing is done using various methodologies and criteria but is based on the Common Criteria testing profile. It is functionally similar to the TCSEC standard notation of D, C, B and A. There are seven EAL levels ranging from EAL1 to EAL7, where testing and verification becomes more in-depth for each ensuing level.

Ethernet

This is a local area networking technology in very wide use. It is simple and effective to use and implement. Today, it can reach speeds up to more than 10 gigabits per second, although 1-gigabit speeds are the most common today. It is defined by IEEE standard 802.3. Older speeds include 10 and 100 megabits per second.

Extensible Mark-up Language (XML)

XML is an open standard for use in data and information interchange. It is a general-purpose mark-up language defined by the W3C. It is highly flexible and can be extended for use by adding additional tags and elements, all of which can be user or application defined.

Free and Open Source Software (FOSS)

Free and Open Source Software commonly refers to software that is free to modify and redistribute according to various licenses that may accompany the software. Specifically, the notion of FOSS requires that the source code be made available. However, FOSS makes no guarantee that software is free of cost or charge.

FreeBSD

This is an open source version of BSD that implements a MAC-based TE control system.

GNU C Compiler Collection (GCC)

Originally written in 1985 by Richard Stallman, it is an open source replacement for modern day programming language compilers, most notably for C and C++ compilers. The full GCC suite supports many languages, currently including C, C++, FORTRAN, ADA, Java, Objective-C and Objective-C++. It is the most widely used and ported compiler suite in existence. Furthermore, almost all open source software has been compiled with it, including Linux and as well as other operating systems.

GNU's Not UNIX (GNU)

GNU refers to an operating system that is composed completely of free software. Officially, GNU does not refer to the GNU/Linux kernel but instead refers to an abstract operating system that uses the GNU Hurd kernel.

How-to

A How-to document is a descriptive text for how to accomplish a given task or series of tasks. It is very commonly found in the UNIX and Linux world (at TLDP). These documents can range in length from short to very long, sometimes long enough to be a short book. These documents are meant to impart knowledge and wisdom on a given subject to non-experts. As How-to suggests, these documents are often like a recipe; follow the instructions and understand the basic concepts and out comes a working result.

Hewlett Packard-UNIX (HP-UX)

HP-UX is Hewlett Packard’s (HP) proprietary UNIX operating system native to the PA-RISC architecture and is used on HP workstations, servers and datacenters. It is based on the SVR4 UNIX standard.

IRIX

IRIX is SGI’s flagship proprietary UNIX operating system. It is a very high-performance system designed for massive parallel computation and supports very large system configurations with thousands of processors and terabytes of memory. Furthermore, OpenGL, designed under IRIX, was an integral part of that operating system.

Information Systems Security Officer (ISSO)

An ISSO officer is an employee or other personnel charged with safeguarding computer networks, computer systems and other telecommunications systems. His responsibility is wide and often includes classified and unclassified networks alike. He is also likely to provide user-based IT security briefing and training, as well as sensitize users to various computer-related security risks. Finally, he will be responsible for deciding which electronic data and computer systems can and cannot be declassified and by which manner declassification will occur.

Java

Java is an object-oriented high-level programming language developed by Sun Microsystems Inc. in the early 1990s. It is a compiled-interpreted language, where the code is compiled into bytecode and executed by the Java Virtual Machine bytecode interpreter. Its syntax is very similar to that of C and C++, although it is not a procedural language. Its greatest advantage is that the Java Virtual Machine and runtime environment have been ported to most of today's popular operating systems, making it as close as currently possible to a universal programming language.

Kernel

The kernel, specifically in the context of the GNU/Linux kernel, is the key component of the Linux operating system. It consists of device drivers, a memory management facility and other computer hardware-based management software that is used to control and regulates the computer system's resources. In addition, it also provides mechanisms such as system calls and inter-process communication constructs for interacting with the system shell which in turn is used by the user to interact with the system in order to get the computer to perform a given task or work. The user never interfaces directly with the kernel. Instead, through the command shell interface or GUI equivalent, the user calls on the kernel to perform some useful work.

Linux

Linux is a free and open source software operating system based on UNIX. The kernel was originally conceived and developed by Linus Torvalds, who holds and maintains the copyright to Linux. With the assistance of thousands of developers around the world, it has turned into a robust, stable and secure operating system, not unlike its other UNIX-based counterparts. Other than the kernel, it is mainly composed of developer-contributed software.

Mandatory Access Control (MAC)

Defined by the TCSEC standard, it is a type of access control restricting access to sensitive data. It attaches labels to all system data and depending on that label; a given user running at a given clearance level may or may not be authorized to work with that data. MAC is an essential component to building trusted systems as specified by TCSEC. The security policy is determined by the system administrator. MAC also forms a required component of Multi-Level Security systems. True MAC implementations are difficult to find and are complex to manage and administrate.

Multi-Level Security (MLS)

MLS is the mechanism that enables computer systems to process data with different security classifications. MLS is implemented using different mechanisms, although the most popular MLS security-enforcement mechanism is Mandatory Access Control. Successful implementation of MLS enables not only processing of classified information, but also its storage.

OpenOffice

OpenOffice is a FOSS office suite that is available for multiple platforms including Windows, Mac OS X, Solaris, Linux and IRIX. It is almost 100% compatible with MS Office file formats and can import them and export to them. OpenOffice's default file format is a compressed XML-based file format.

Operating System

An operating system can be defined as a collection of binary executable code and system configurations separated into individual computer files and that controls the computer's hardware, operating system behaviour and user-based applications, tools and utilities. An operating system is comprised of a kernel, a shell or graphical user interface for interacting with the kernel and launching system and user-based applications.

National Security Agency (NSA)

NSA is the American cryptographic, communications and computer security organization performing its functions on behalf of the U.S. government.

National Institute of Standards & Technology (NIST)

A part of the Department of Commerce, its objective is to propagate and advance American scientific standards and technologies. Specific divisions of NIST work in close collaboration with the NSA to maintain IT security standards including the CC certification and accreditation.

Network

A network is a conglomeration of computers and other network-enabled devices that use standard communication media (e.g. Ethernet) as a means of interoperating with one another. A network does not have to be made up of the same type of systems. Heterogeneous networks are very common. If a system is connected to a local network but is unable to communicate with other systems because it is not configured to do so, then it is normally not considered as a part of the network even if it is attached to it. It is not sufficient to be connected to a physical network to be a part of it, as a system must also be configured to participate in that network

Polyinstantiation

In the world of MLS and TCSEC, polyinstantiation generally refers to the ability to spawn multiple instances of the same applications and services for the same user at different security levels. Each instance in which a user works or uses generates a new instance at that level of security. Additionally, polyinstantiation also refers to multiple instances of disc data including files and directories, each of which is only accessible when a user is using a given security level.

Portable Document Format (PDF)

Developed by Adobe Systems, PDF is an open file format specification for document and data interchange that is independent of its editing software, printer drivers or underlying operating system or platform. A PDF document will appear the same on any computer system.

Principle of Least Privilege

The principle of least privilege states that a user, application or service is to be given the least access necessary to system resources, permissions and privileges in order to perform a given set of tasks. It is often required that the system administrator profiles the various applications and services to be used and accords them with the necessary privileges to carry out their tasks.

Privilege Escalation

Privilege escalation is commonly referred to in computer circles as the successful exploitation of a bug or vulnerability that results in an attacker gaining additional system privileges such as root shell. However, TOS systems often provide a mechanism for a user to gain additional rights, permissions and privileges in order to carry out some tasks. Privilege escalation in this sense is valid and appropriate permissions must be assigned to the user by the system administrator or security administrator.

Programmable Read-Only Memory (PROM)

PROM, commonly referred to as EEPROM (Electronically Erasable Programmable Read-Only Memory) is a computer chip that holds volatile configuration data that can be rewritten, reprogrammed or erased. It is commonly used to store boot parameters and administrative passwords.

ProPolice

Available as a patch to GCC, ProPolice offers a method for enhancing and securing an application's stack, thereby making it less vulnerable to buffer overflow and stack-based attacks.

Proxy Web Server

This type of server caches Internet-based requests. Proxy web servers generally find themselves on routers, gateways, firewalls or other devices connected to or near an Internet connection.

Revised Interconnection Protocol Security Option or Revised Internet Protocol Security Option (RIPSO)

RIPSO is similar to CIPSO in that it too is a TCP/IP-specific security label-based encapsulation protocol used for the open exchange of classified information between different TOS systems. However, unlike CIPSO, the security label of the packet is not based on the security label of the targeted data. Instead, the sensitivity label of the packet must be manually specified.

Role-Based Access Control (RBAC)

RBAC is an alternative technology for restricting computer system access. Instead of users directly gaining access to administrative accounts at login, the user must “switch” to an administrative role. Users login to the system as they normally would and if they have permission to assume one or more administrative roles, they then switch to a specific role. By assuming the role, the user then has the necessary system privileges to carry out tasks that would normally be done using that specific role. RBAC’s objective is to restrict the usage of administrative accounts and break these accounts up according to the tasks that are to be performed. The security advantage is twofold. The first is that the role cannot be accessed from outside the system, as an actual login and role assumption are required, preventing illicit use of available network services and applications. Secondly, it defines which users can perform specific administrative roles. RBAC, first implemented under UNIX, is now available for many other systems including Linux and BSD.

Root

Root refers to the system super user that is generally the system administrator. Root is a special account that is used to perform system administration related tasks. Most modern operating systems have similar accounts even if they are not named Root.

Runlevel

A runlevel describes the system state of the underlying computer system. Generally, runlevel are used to denote the system state of the underlying system, which is commonly a UNIX-based operating system. Under Solaris, runlevels range from 0 to 6, where 0 indicates that the system has halted or is paused at the BIOS. Runlevel 1 indicates that the system is booted in single-user mode. Runlevel 2 typically indicates multi-user mode and runlevel 3 indicates multi-user mode with networking and graphics. Runlevel 4 is left for user-based configurations and Runlevel 5 shuts down the system. Finally, runlevel 6 is reserved for system reboots.

Sanitization

Sanitization refers to the wiping or scrubbing of data from electromagnetic media including floppies, hard disk drives and USB devices in a manner that writes multiple patterns over the same sector in order to permanently expunge any electromagnetic evidence of previously existing data. Sanitization also refers to the editing and redaction of electronic documents whereby highly classified portions are permanently expunged in a secure and permanent manner so that the classification of the underlying file can be lessened or altogether removed.

Security-Enhanced BSD (SEBSD)

SEBSD is based on SELinux. It uses the same security architecture, framework and mechanisms as SELinux in order to provide Mandatory Access Control. It is not yet ready for enterprise deployment and is still in development.

Security-Enhanced Linux (SELinux)

SELinux is a set of modifications/improvements to the Linux kernel made public by the National Security Agency and that can implement Mandatory Access Control on a system.

Solaris

Solaris is the proprietary UNIX operating system for Sun Microsystems Inc. It is a powerful and highly robust operating system and is arguably the most popular commercial UNIX platform available. It is known for its tight integration of Java technologies and for its excellent administrative tools. It is available for both PC and SPARC-based platforms.

Scalable Processor Architecture (SPARC)

The Scalable Processor Architecture is a high performance open processor architecture developed by Sun Microsystems in 1985. It is based on RISC technology. Today, it is a 64-bit computing platform and is in use as a microprocessor by a consortium of companies including Sun Microsystems.

Stack smashing

Stack smashing is the process whereby an application or system stack is smashed, penetrated, or exploited due to a known or unknown bug or vulnerability. Buffer overflow attacks are a common example of stack smashing attacks.

Switch user or Substitute User (Su)

Su is a UNIX based command used to assume the role or shell of a given user. That user may be a standard system account, system administrator, service or daemon. The Su facility is commonly used by users to gain system administrative access without having to logout from their account.

Superuser Do (Sudo)

The Sudo mechanism is a system facility that allows users to execute superuser (root) commands. However, Sudo requires that a user have the appropriate security privileges in order to execute the facility, otherwise the system will prompt for a password. The exact functionality of Sudo will be highly dependent on how it has been configured.

Transmission Control Protocol/Internet Protocol (TCP/IP)

The Transmission Control Protocol/Internet Protocol is a suite of protocols developed by DARPA to facilitate and enable communications between different hosts and different networks. Data is transmitted using network packets. However, the encapsulation of these packets can be achieved in various ways. Internet Protocol (IP) is a connectionless protocol that makes no guarantees about the reliability of the transmission of the data. User Data Protocol (UDP) is another connectionless protocol and is a part of IP protocol suite. Network routing is possible because the IP protocol is itself routable as an IP packet contains all the necessary information within its header for successful routing. The Transmission Control Protocol (TCP) is a connection-based protocol that provides multiplexing and is more reliable than IP because TCP makes reasonable efforts to ensure that data packets are received by the destination without corruption. TCP is also a part of IP. TCP/IP is the modern standard for networking and is found in most modern networks, as well as throughout the Internet.

TrustedBSD

See SEBSD.

Type Enforcement (TE)

Type Enforcement is a fine-grained security access-based control mechanism. Type Enforcement provides MLS capability through MAC, which in turn uses object labels to determine how objects are to be handled. Type Enforcement implements and defines the security rules pertaining to how MAC handles labelled objects.

Trusted Computer System Evaluation Criteria (TCSEC)

The Trusted Computer System Evaluation Criteria is also known as "The Criteria" or "Orange Book." This document is published by the National Computer Security Center, commonly referenced as DoD document 5200.28-STD. TCSEC is a series of requirements and evaluation classes that outline the security effectiveness of hardware and software systems. These criteria are designed to evaluate systems that will process/store sensitive, protected or classified information.

Trusted AIX (Advanced Interactive eXecutive)

A high-performance, robust, stable and flexible, TCSEC (Orange Book) and CC certified UNIX operating system, based on the standard AIX distribution. AIX and Trusted AIX are developed by IBM and can be run on IBM PowerPC systems. See AIX.

Trusted HP-UX (Hewlett-Packard-UNIX)

A high-performance, robust, stable and flexible, TCSEC (Orange Book) and CC certified UNIX operating system, based on the standard HP-UX distribution. HP-UX and Trusted HP-UX are developed by HP and can be run on both the Alpha and Itanium platforms. See HP-UX.

Trusted IRIX (TRIX)

A high-performance, robust, stable and flexible, TCSEC (Orange Book) and CC certified UNIX operating system, based on the standard IRIX distribution. Both distributions are native to SGI-based platforms.

Trusted Networking

Trusted Networking is an operating system-based networking security mechanism by which only trusted computer systems could communicate with one another and exchange both classified and unclassified information via the network. Trusted Networking makes changes to the TCP/IP protocol stack by implementing security context labelling directly into the various packets. In addition, Trusted Networking allows the system and security administrator to fine-tune network access to the outside world, both for the system as a whole and on a per user basis.

Trusted Operating System (TOS)

This is generally a modified UNIX-like operating system that incorporates many or all of the features outlined in TCSEC and has been certified by a TCSEC certifying or Common Criteria (CC) certifying authority. Generally, an operating system is said to be trusted when it achieves a TCSEC rating of B3 or better, or CC EAL3+ Controlled Access Protection Profile (CAPP) or better.

Trusted Path

The Trusted Path is a communication-based mechanism that helps to ensure that a user is in fact communicating with the intended communication channel. More specifically, as part of the Trusted Computing Base put forward by Orange Book TCSEC requirements, it requires that the system provides a visual feedback to the user indicating that the user is in fact interacting with a trusted system component. These trusted components are used for logins, security contexts, data downgrading, etc. Without the visual feedback, the user is not able to distinguish trusted from non-trusted components of the system desktop. The Trusted Path forms an integral component of the TOS desktop environment.

Trusted Solaris (TSOL)

A high-performance, robust, stable and flexible, TCSEC (Orange Book) and CC certified UNIX operating system, based on the standard Solaris distribution. Solaris and Trusted Solaris are developed by Sun Microsystems and can be run on both PCs and Sun SPARC-based systems. See Solaris.

UNIX

A multi-user, multi-tasking, multi-threaded operating system based on a kernel that provides a consistent interface to the user for both interactive and background job processing. UNIX is multi-platform and hardware independent and supports advanced APIs. It is generally considered by the computing industry as the hallmark of scalable, robust, secure and reliable computing. It was originally developed at AT&T Labs by Dennis Ritchie and Ken Thompson.

Windows

Windows is a graphically based operating system developed by Microsoft. First introduced in 1985, it has undergone many changes and iterations since then. Now, it is a multithreaded, multiprocessor, multitasking operating system and it is the most popular desktop operating system currently in use worldwide. Due to its prevalence, it is also the most heavily attacked operating system worldwide. Because the operating system must cater to a very large variety of computing platforms and hardware peripherals, it lacks certain security features found in more hardware-specific operating system implementations. However, great strides have been made over the recent years to improve its security. Past iterations have included Windows 3.x, 95, 98, NT, ME, 2000, XP, 2003, Vista, 7 and 8.

Write Up – Read Down (WURD)

Based on the Bell-LaPadula Model, Write Up – Read Down pertains to the flow of data given an MLS system. Based on the model, a user can read data at an equivalent or lower security context than he himself is currently using. However, that user cannot write to files at a lower security context than he himself is currently using as classified data from the currently higher security context could cross-contaminate data at a lower security context. Therefore, if data is written to a file of lesser classification, that file must have its security context upgraded to avoid data and classification contamination related issues.

This page intentionally left blank.

DOCUMENT CONTROL DATA (Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)		
1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.) Defence R&D Canada – Valcartier 2459 Pie-XI Blvd North Quebec (Quebec) G3J 1X5 Canada		2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.) UNCLASSIFIED (NON-CONTROLLED GOODS) DMC A REVIEW: GCEC JUNE 2010
3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.) Installing and configuring a declassification system: A solution combining Trusted Solaris and Free and Open Source Software		
4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used) Carbone, R.; Vincent, S.		
5. DATE OF PUBLICATION (Month and year of publication of document.) February 2013	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.) 66	6b. NO. OF REFS (Total cited in document.) 76
7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.) Technical Memorandum		
8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.) Defence R&D Canada – Valcartier 2459 Route de la Bravoure Quebec (Quebec) G3J 1X5 Canada		
9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)	
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.) DRDC Valcartier TM 2009-086	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)	
11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.) Unlimited		
12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.)) Unlimited		

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

(U) In spring 2008, certain project requirements necessitated the use and implementation of a system that could be used to declassify classified data for use on unclassified systems and networks. Some of the researchers at Defence Research Development Canada – Valcartier wished to be able to share declassified versions of their work with others. However, data residing on classified systems and networks cannot easily be transmitted to unclassified systems. Although it can be done, a variety of safeguards, data containment and compartmentalization and various procedures must be followed for doing so. The challenge is not so much the procedures and safeguards, but rather using an accredited system for containing and compartmentalizing the data to be declassified. Since no DND policy is clear on which systems to use or the process by which data can be declassified, the primary author decided that by combining commercially available software with specific FOSS components, a secure system could be built that would satisfy even the most stringent security requirements. The solution chosen was a combination of Trusted Solaris and various commonly used FOSS packages. This memorandum therefore describes how such a system is built and configured.

(U) Au printemps 2008, certaines exigences de projets nécessitaient l'utilisation et l'implantation d'un système qui pourrait être utilisé pour déclassifier des données classifiées pour une utilisation sur des systèmes et des réseaux non classifiés. Certains chercheurs à Recherche et développement pour la défense Canada – Valcartier voulaient être en mesure de partager des versions déclassifiées de leurs travaux avec d'autres. Toutefois, des données résidant sur des systèmes et réseaux classifiés ne peuvent pas facilement être transmises sur des systèmes non classifiés. Même si cela peut être fait, un éventail de protections ainsi que différentes procédures pour le confinement et le cloisonnement des données, entre autre, doivent être suivies pour y parvenir. Le défi n'est pas tant les procédures et les protections, mais plutôt l'utilisation d'un système accrédité pour le confinement et le cloisonnement des données à être déclassifiées. Puisqu'aucune politique du MDN est claire sur les systèmes à utiliser ou le processus par lequel les données peuvent être déclassifiées, l'auteur principal décida qu'en combinant des logiciels commerciaux disponibles avec des composants à code source libre et ouvert, un système sécuritaire qui satisferait même les exigences de sécurité les plus strictes pourrait être construit. La solution choisie est une combinaison de Trusted Solaris et de différents progiciels à code source libre et ouvert couramment utilisés. Ce mémorandum décrit donc comment un tel système est construit et configuré.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Common Criteria; Declassification; FOSS; Free and Open Source Software; MAC; Mandatory Access Control; MLS; Multi-Level Security; Orange Book; Privilege escalation; Privilege separation; RBAC; Role Based Access Control; SEBSD; SELinux; TCSEC; Trusted Operating System; Trusted Solaris; TSOL