TURNING MULTIPLE EVALUATED PRODUCTS INTO TRUSTED SYSTEMS

NCSC TECHNICAL REPORT-003

Library No. S-241,353

July 1994

FOREWORD

This Technical Report "Turning Multiple Evaluated Products Into Trusted Systems," is provided to stimulate discussion on how evaluated products can be combined to produce trusted systems. We establish the premise that the integrator/system designer has the responsibility to retain, in as much as possible, an evaluated product's rating while it, the product, is performing within the context of the integrated (larger) system. In this manner, we therefore propose that a modified evaluated product has advantage over the use of a non-evaluated product for similar functionality.

Recommendations for revision to this publication are encouraged and will be reviewed periodically by the NCSC. Address all proposals for revision through appropriate channels to:

National Computer Security Center
9800 Savage Road
Fort George G. Meade, MD 20755-6000

ATTN: Standards, Criteria, and Guidelines Division

*Reviewed by:*____
GLENN GOMES
Chief, INFOSEC Standards, Criteria & Guidelines Division

*Released by:*____
ROBERT J. SCALZI
Chief, INFOSEC Systems Engineering Office

ACKNOWLEDGEMENTS

This document was written by Joan Fowler and Dan Gamble of Grumman Data Systems for the Procurement Guideline Project. The project leader was MAJ (USA) Melvin L. De Vilbiss. Besides many NSA organizations, the document was reviewed by Department of the Army (ASIS), DISA, MITRE, and NAVELEXSECSEN.

TABLE OF CONTENTS

5. CONCLUSION

BIBLIOGRAPHY

1 INTRODUCTION

In the past few years, more Commercial Off-The-Shelf (COTS) products have been populating the Evaluated Products List (EPL) than in previous years. In the current economic environment, the tendency is to use evaluated products when designing trusted systems to meet specific procurement requirements. The process to design a trusted system composed of evaluated products is fundamentally the same as designing any system using COTS products. The concept that makes the process of designing trusted systems unique is that the combination of different products composes a totally new security environment.

A trusted system, in the context of this paper, is a system composed of multiple products. This system, at the interface to the Trusted Computing Base (TCB), conforms to the Department of Defense (DoD) Trusted Computer System Evaluation Criteria (TCSEC) (DoD 5200.28-STD) [1] and the forthcoming TCSEC-derived protection profiles to be embodied in future U.S./international criteria.

This paper discusses how evaluated products can be combined to produce trusted systems which meet the requirements specified in a procurement document, thereby modifying, adapting, or eliminating portions of the composing product's TCB. Frequently, the requirements specified necessitate changes to the product TCBs. Because the product's rating may be invalidated when the product's TCB is changed without understanding, justification, and review; system-level assurances are necessary to compensate for the changes. It is the responsibility of the system integrator/system designer to do the utmost to retain and not invalidate the product rating. However, even with this possible invalidation, the use of an evaluated product in a system provides the knowledge that the original product was scrutinized, and those portions of the product that are not changed continue to retain that scrutiny for the correctness of processing. Therefore, even if a product's TCB must be modified, adapted, or portions eliminated, the use of an evaluated product in a system development is advantageous over the use of a non-evaluated product for the similar functionality. The combination of unequal security qualified components to build a system is another dilemma in the integration process which will not be discussed in this paper.

The need for the modification, adaptation, or elimination of a TCB in evaluated products has greatly diminished in recent years. When the modification, adaptation, or elimination is dictated due to system requirements, these changes can take many forms. The easiest and most trusted form is to tune the product using the product's configuration options, "hooks", or switches. (For example, in many products it is possible to audit all or no activity for a user.) Another form is to use the product as it was not necessarily intended to be used. If a

product with Mandatory Access Control (MAC) labels and controls is used in a system high environment, the MAC processing actually occurs in the execution of the software, but it does not have any security relevancy in the system. Another form of adapting an evaluated product's TCB is to develop an extension to overcome the shortcomings of the combined products used in a system. A final form of eliminating security functionality is to actually modify the code of the product. This form is the least desirable and should only be done when the system requirements dictate that product code modification is the only solution. No matter which form the modification takes, great care must be taken to determine the effect on the entire system. The time required to integrate evaluated products into a trusted system and ascertain the effects on each facet of the product must be assessed since that time, in some cases, may be greater than the time required to develop a trusted system, or a portion of a trusted system, from the beginning.

2 SYSTEM DESIGN APPROACH

This section discusses an approach to designing a system to integrate COTS products. This approach is a single method that can effectively be used for system integration, although it is not the only approach. The approach, as it is described in this section, is used for the integration of untrusted systems from COTS products. It is also applicable to the integration of EPL products into trusted systems, with a few modifications to the approach. This revised approach for trusted systems will be discussed in following sections.

2.1 CLASSIC HIGH LEVEL VIEW OF A SYSTEM

The textbook high level view of a system is a processing box which receives inputs, processes the inputs according to a set of requirements, and generates outputs. This is the high level view of a system whether it is trusted or not. The list of requirements which must be satisfied by the system processing is defined by the operational needs and outputs required of the system. In the case of trusted systems, the security policy of the system also determines some of the system requirements. All of these requirements may be defined in a Request for Proposal, a System Specification, a Statement of Work, or some other type of requirements document. Finally, these requirements must be available to the system integrators/designers for analysis and subsequent design of the system.

2.2 DETERMINE SYSTEM FUNCTIONS/SERVICES

When designing a system, the first step beyond this classic high level view of a system is to determine what functions must be performed, as defined by the requirements for the system.

A function is a "series of related activities, involving one or more entities, performed for the direct, or indirect, purpose of fulfilling one or more missions or objectives. It should be identifiable and definable, but may or may not be measurable." A function may be composed of one or more subfunctions. [2] Subfunctions perform a portion of the overall task assigned to the function.

Each function selected for the system should be internally cohesive in that it performs a single task and requires little interaction with other functions in the system. [5] Another objective in determining the functions is to minimize coupling between the functions to make them as independent as possible. [4] Of course, no system can exist without some coupling to preserve the cohesiveness of the system as a whole. By definition, a function that is not bypassable becomes primitive within an architecture. That function's implemented security

policy will be invoked between each domain that it invokes. Unintentional or intentional emergent behavior can be created when integrating functions which detract from the cohesiveness of the system functionality.

Some examples of high level functions that may be determined for a system are data base management, man-machine interface (MMI), communications, or mail. In trusted systems, MAC, Discretionary Access Control (DAC), Audit, and Identification and Authentication (I&A) are all possible functions to be defined. The definition of any or all of these functions is determined by the set of requirements for the system. There are security requirements that are not normally characterized as functions. Examples of these are domain isolation, integrity, and trusted path. However, if a system or product has a trusted path available to the user for example, some mechanism (e.g., "function") must provide this capability.

2.3 DEFINE FUNCTIONS/SERVICES INTERDEPENDENCIES

The next step toward designing a system is to determine the coupling that has to exist between functions. This coupling forms interdependencies between the functions or services. An example of this interdependency at a high level is a mail function that may be dependent on the MMI to "deliver" the mail to a user's terminal. Of the security functions, applications may be dependent on the TCB to perform security functions. Additionally, the I&A function may need the MMI to allow the user to input his/her logon identification sequences. Finally, the MAC and DAC functions depends on an I&A function to authenticate and provide the correct information for the user.

2.4 SPECIFY DEPENDENCY LATTICE

Once all of the functions have been defined and the interdependencies have been determined, a dependency lattice can be described. **Figure 1** illustrates a dependency lattice for generic functions. This lattice defines those functions that are dependent on other functions, as well as those functions that are independent.

2.5 DEFINE PRODUCTS AND PLATFORM

Finally, the independent functions are used to determine, from the products available, those products that will best meet the requirements of the system. This is done by comparing the functions required by the system with the functionality provided by all the available products. When a close match is determined, a product can be selected. Sometimes dependent functions have to be rearranged to better fit the products that are available. There is never a perfect match between the requirements for a system defined into functions and the specifics of a single product or a group of products. The products will either not collectively contain a needed dependent function, will contain functions that are not requirements for the system, or will contain redundant functions among the group of products.

Once the best correlation between all the functions or services and available software products is made, then the physical requirements are taken into account. These physical requirements include performance, reliability, interfaces, and other requirements [5] which further constrain the choice of available software products, and thus determine the platform (e.g., hardware) for the system. Again, there is never a perfect map between the software products selected, the system's physical requirements, and the platforms available even when the platform is selected at the end of the process. However, selecting the platform prior to determining the software products that will satisfy the system requirements increases the differences between the map of the platform and the products and physical requirements.

3 TRUSTED SYSTEM DESIGN APPROACH

The approach to the design of trusted systems using evaluated products must be taken a step further than the approach described above. When designing trusted systems, the security functionality of each individual product may not satisfy all of the security requirements of a system. For instance, one product may have a compliant I&A (e.g., with an automatic password generator), while another product may have a compliant audit mechanism (e.g., with all of the reporting capabilities for the audit log). However, the security functionality of all of the products together may present a redundant surplus of security functionality. Redundant security functionality is especially important to deal with when there are conflicts between the security functions of the various products to be used for the system. A possible example of a conflict is the case of object reuse functions in a system in which one product clears objects before releasing the object to the user, and the other product in the system clears the object after the user has released the object. In this case, the potential exists for the user to receive, under the right circumstances, an object that has not been cleared by either product; or the user may suffer performance degradation when the object is cleared by both. In this case, a unified object reuse policy for the system would need to be established.

3.1 EVALUATED PRODUCTS LIST (EPL) PRODUCT DETERMINATION

An evaluated product is selected much as any other product would be selected, based on a set of functions that the product must satisfy. As stated above, when a function and its dependent functions are compared to a product, there are almost always requirements that are not satisfied by the product. Additionally, there is functionality in the product that is not included in the list of requirements for the system as a whole. This surplus may lead to conflict between products when each attempts to satisfy the same single requirement in a system with a cohesive policy.

An example of this conflict is a single processor system that has requirements translating into a need for an evaluated operating system and a trusted application, (e.g., mail). The operating system will probably contain I&A, DAC, and audit capabilities. The application may also have I&A, DAC, and audit capability. In all other aspects, the two products are a perfect match for the system requirements. However, in this case, there is a redundancy of security functionality. The application is not an operating system and the operating system can not perform the non-security capabilities required of the application. Therefore, neither of the products individually satisfies both the security and non-security requirements of the system. If two products with a reference monitor are included in a system, one of the reference monitors is going to be bypassed at some time during operation of the system.

The redundant features issue can be decomposed into security policies and mechanisms to implement the policy. If both the policy and the mechanism are identical, as in the case of a homogeneous network environment with a single policy in which the workstation and server both use the same evaluated operating system, then there might be user resistance (e.g., to a double logon). If the same intended policy is implemented with different mechanisms, as in the case of a heterogeneous network environment in which two different operating systems are used with different labeling schemes, then there exists a conflict between the two mechanisms. The label conflict may be resolved by a conversion function developed as an extension to the TCBs of either or both of the products. Additionally, if the policy is different but the same mechanism is used, a policy conflict exists even in a homogeneous workstation and server environment with the same operating system containing the same DAC mechanism. The workstation may be using different "Group" definitions and Access Control Lists (ACLs) than the server. This conflict would violate one of the policies without the knowledge of the violated processor. Finally, if both the policy and the mechanism are different, a heterogeneous network environment in which the label policy and the labeling mechanism are both different, then conflicts that might not be able to be resolved may exist. In this case, something fundamental in the policy or the mechanism would have to be changed. The simple conversion of the label format would not suffice to integrate these two systems.

3.2 PRODUCT CONFLICT RESOLUTION

It is not efficient to have differing DAC or audit schemes when designing a cohesive system. This is not to state that redundancy can not, in some circumstances, strengthen the security of a system, provided that it is user friendly and not counter to human intuition. However, there is always a concern for consistency of the global security policy of the system where redundancy is involved. It is not advantageous to incorporate two I&A mechanisms into a single secure operational system, without at least some dominance of one over the other. (Most systems today require a limiting of a single logon for a user session.) Each of the redundant security functions may need to be modified or disabled in one of the products (through extensions to the product TCB, switches, configuration options, if possible; or TCB code modifications, if necessary) in order that the system may have a single I&A, DAC, or audit. This is done by modifying, adapting, or eliminating one or the other product to disable or limit the function. Then the other product, in which the function is not disabled or limited, must be changed to interface with the product in which the function has been disabled or limited.

The process of modifying or adapting an evaluated product by limiting functionality has ramifications to the evaluation of the modified product. It may invalidate the EPL rating of the product, if not done with review, justification and understanding. The integration of multiple evaluated products may stay within the bounds of the assumed parameters as stated in the individual product's evaluation report, or the integration effort may violate those bounds. If necessary, it is the responsibility of the system integrators/designers to compensate for any invalidation of the product rating using system-level, as opposed to product-level, assurance. This will be discussed later in this paper.

The product vendor is always the best choice to make modifications to products. The vendor may make a business decision on the marketability of changes required for a system acquisition. If the modification can be productized, the vendor will insert the change into the standard product and perhaps take the modification through the Rating Maintenance Program (RAMP) frequently with no charge to the acquisition. This is the most advantageous course of action. The spectrum from the above (vendor made changes) down to the integrator performing the changes without any vendor support are possible scenarios. The average contract design, integration, and/or development strategy will lie somewhere along this spectrum.

3.3 ARCHITECTURE RELIES ON EXTERNAL DEPENDENCIES

Frequently, there are external dependencies which affect the architecture of a trusted system which would not affect the architecture of an untrusted system. An example of this is a system that receives labeled input. This system receives the labeled input directly into the processing stream for all data. Since the input is labeled at the source of the data outside of the system boundary, the integrity of the label must be assumed to be trusted as far as the system is concerned. (Mechanisms are available to ensure this to be true.) Therefore, the MAC performed using this label is solidly based.

However, if the data received by the same system architecture is not labeled and is at multiple classification levels, then the system does not have a basis for MAC. The architecture could be changed to include some sort of labeling entity prior to the unlabeled data entering the mainstream of the system. Depending on the requirements of the system, this could be a human on a terminal reviewing and labeling all data; it could be a front-end component labeling all data from a single level device; or, it could be an operating system labeling all data from a single level port. For this example, it does not matter what the architectural change would be, just that the overall system architecture must accommodate the differences between labeled and unlabeled input.

3.4 TRUSTED COMPUTING BASE (TCB) DEFINITION

Once the products are selected and the architecture is defined, the TCB for the system must be established. Under the premise of this report, the system would be designed using COTS components (both trusted and non-trusted products). A single system TCB would, in this case, be defined using the product TCBs as the basis and satisfying the reference monitor assumptions and the system security policy. This is done by examining the various TCBs of the products, identifying the mechanisms and interfaces that will remain for the resulting system, and analyzing what additional mechanisms and interfaces may be necessary for the system.

3.4.1 Product Analysis

As stated previously, there is never a perfect match between requirements, functions, and products. If functionality is lacking in all of the products selected, then the integration process must include the development of that functionality or the inclusion of a non-trusted component to handle the functionality. Occasionally during a tradeoff analysis, a non-automated solution (e.g., a locked room) is determined to be the preferable manner to address any missing functionality.

However, the more likely occurrence, when a collection of evaluated products are combined, is redundant security functionality. An analysis must be made to determine which security features will be used in each product. This analysis must be carried a step further for evaluated products. An additional analysis must be made to determine how the security characteristics of each individual component may affect the composite characteristics of the system, and what the resulting effect will be to the overall product and system when a product's security feature is not used, either disabled or limited. It is important that this analysis be performed in the early stages of a program to inform the program management of the correct integration options, even if the demonstration/proof of the satisfaction of the requirements of the TCSEC by the modified system is required for the integrated system. To rely on the later assurance proof for this analysis will inform the program, after delivery, that the system has already been integrated/developed incorrectly. At that point, the information is not beneficial to the program.

3.4.2 System Interface Analysis

Beyond the analysis of the product and the selection of which product features to use and not use, a system analysis must be performed to identify the interfaces that will be needed within the system TCB. This analysis includes the system interfaces that will occur between the products without modification, or as the manufacturer delivered it. (Again, the use of the code of products as they are delivered by the manufacturer is the preferable manner in which to use a product.) Additionally, the analysis must take into account the interfaces which are newly created when the products are modified to eliminate certain features, or add a system capability.

A desirable result of any trusted system integration is to minimize the overall system TCB while minimizing the impact on the product TCBs composing the system TCB. Using evaluated products, each will contain a TCB. When all of the product TCBs (as well as the new TCB functions developed for the integration effort) are taken into account for the system, the resulting overall TCB will be a certain value. To eliminate a portion of a product's TCB is to diminish the size of the overall system TCB by the complexity and value of the portion of the product TCB that is eliminated. This serves to minimize the overall system TCB by the value of the excluded portions of all the products' TCBs. However, this minimization action must be accomplished with care. Eliminating parts of a component TCB may increase your risk because of internal dependencies within the product. Additionally, it may increase program cost because the impact of removing the portion of the product TCB must be determined. **Tradeoffs and compromises must be made.**

3.4.3 Application Audit Example

Figure 2 is a pictorial description of the audit function of a trusted application. The application could be anything trusted, a trusted mail application, a trusted Database Management System, etc. This particular audit function has a security administration subfunction which sets the criteria on which auditing will occur. The criteria are placed in a database. The next subfunction is the audit interface in the TCB which detects a criteria match. When a match is detected, the event recorder subfunction records the event using the user ID, success/fail criteria, event data, and time which are held for the application in a database, table, or global common, depending on the implementation. The event recorder writes the audit record to the application's audit log. There is also a real time subfunction which checks thresholds and responds to the matching of these thresholds. An example of this functionality is a limit of three attempts to logon using a single user ID. On the fourth attempt, the real time subfunction may lock a user out of the system. There are also several administrative subfunctions dealing with the application's audit log. The data reduction subfunction handles the queries and responses to the audit log. The administrative subfunction allows an administrator to archive and purge the audit log.

3.4.4 Example for An Integrated System

To carry on with this example, the following is a single approach to use this product in an integrated system. (This approach is not the only approach that can be used, neither is it meant to be a procedural description of composing systems.) The product has been selected to perform whatever application it does. In this example, the product will be used in a distributed architecture which has a requirement for centralized administration of the auditing capability and a centralized system audit log. This is not to imply that an application's audit log must be deactivated if there is a system audit log.

Figure 3 illustrates the system with centralized audit administration and storage. The application described in the previous subsection is in the figure as the lightly shaded large box. In order to achieve centralized administration, an audit management subfunction must be developed that sets the criteria for the entire system. A portion of this subfunction must be written to interface with the security administration subfunction of the application. To have the application's event recorder subfunction write the audit records to the system audit log instead of the application's audit log, a common interface must be written between the event recorder subfunction and the system audit log. Assuming that there is more than one application in the system which produces audit records, the common interface subfunction would translate all of the application audit record formats and data packing schemes to a single system audit record format. Additionally, the interface between the application event recorder subfunction and the application's audit log must be severed.

As can be seen from **Figure 3**, there are two new subfunctions in this system view, the audit management and the common interface. These subfunctions are denoted in the boxes without shading. There are also three new interfaces. In the figure, these interfaces are denoted by the heavy arrow lines. There is a new interface between the new system audit management subfunction and the application security administration subfunction. There is another new interface between the application event recorder and the new common interface subfunction. And, finally, there is a new interface between the new common interface subfunction and the system audit log.

Since all of the audit records are now being processed into the system audit log, the application's audit log is no longer used. Therefore, the interface between the application event recorder subfunction and the application's audit log is severed. This is designated in the figure with a heavy "**X**". Finally, since the application's audit log is no longer used, the three subfunctions that support the application's audit log are also no longer needed. These three subfunctions (data reduction, real time, and administrative) and the application's audit log are all designated in the heavily shaded boxes.

4 TRUSTED SYSTEM ASSURANCE

The use of evaluated products is an extremely good starting point for the certification and accreditation efforts of systems. However, the combination of evaluated products, with the resulting changes to the products as described above, may invalidate the rating of the product when the changes are performed without the proper review and understanding. The assurances developed at the system level during the integration process must compensate for any invalidation of the product rating.

The TCSEC is the standard used to develop the assurance of products. The TCSEC defines the assurance documentation required for a TCB. The design documentation requirements are a subset of the overall documentation described in the TCSEC. The TCSEC requires that "If the TCB is composed of distinct modules, the interfaces between these modules shall be described." [1] This is true for all classes defined in the TCSEC above the Minimal Protection Division (D). Additionally, the TCSEC requires that "The specific TCB protection mechanisms shall be identified..." [1] This is a requirement for all classes in the Mandatory Protection Division (B) and Verified Protection Division (A).

Of course, there are additional assurance documentation requirements that include: a security policy model, a Philosophy of Protection, a Descriptive Top Level Specification, a Formal Top Level Specification, a covert channel analysis, a TCB verification report, a Configuration Management Plan, administrator and user manuals, and testing documentation. The modification, adaptation, or elimination of product TCB functionality (mechanisms and interfaces) has a ripple effect through all of the assurance documentation for the system.

Security testing, as well as other activities such as architecture, recovery, and verification, are also required as assurance mechanisms. Security testing of the combined evaluated products demonstrates that the modified mechanisms and interfaces perform as intended and that the overall level of protection has not been diminished. Finally, this testing will serve to validate the completeness of the system level documentation. Security testing of the system, as with all assurance activities, is performed to support a certification and accreditation, and not an evaluation, of the system. All the engineering efforts to assure a system are documented (e.g., security testing is reflected in the test plan, procedures, and report required by the TCSEC for testing). Therefore, the remainder of this paper uses the term "documentation" to refer to all of the assurance documents required by the TCSEC for evaluation. Included in the use of the term "documentation" are all the activities (e.g., testing, design engineering, covert channel analysis) that are performed in order to produce these assurance documents.

4.1 PRODUCT ASSURANCE DOCUMENTATION

In order for a product to be evaluated, TCSEC documentation requirements have to be satisfied. But what happens to this product assurance documentation when the product is modified for use in a system? Most of the product documentation should still be valid. If the product changes so much that a total rewrite of the documentation is needed, then perhaps the product is not really a match for the requirements of the system, and another product should be selected.

4.2 SYSTEM ASSURANCE DOCUMENTATION

Assuming that most of the product is going to be utilized as evaluated in the system, and that most of the product's documentation is therefore valid, the few modifications, adaptations, and eliminations made to the product must be documented. When composing evaluated products into trusted systems, new subfunctions may be needed to couple products, new interfaces are included to these new subfunctions, some of the mechanism of the original product may be disabled, and original interfaces may be excluded. These four types of modifications break down into two categories: TCB interfaces and mechanisms. The modifications are the two sides of each of these categories: eliminated and new TCB interfaces; and eliminated and new mechanisms.

The existing evaluation version of the product documentation should describe all interfaces and protection mechanisms to include both the original interfaces and mechanisms that have been eliminated during the integration of the system. The system level documentation should describe the effect that the elimination of the mechanisms and interfaces of the evaluated product has on the system TCB as a whole.

The previous paragraph covers the elimination of original interfaces and mechanisms of the evaluated product used in the system. The addition of new mechanisms and the resulting additional interfaces to the combined product TCBs for the system must also be documented in the system-level assurance documentation. These mechanisms and interfaces are not described in any of the product-level documentation since they are probably either not available in any of the individual products, or were not required to perform in the product as they are in the system.

There are options to the system integrator/developer when the modification of product documentation is done. The vendor may develop the code modifications and document those modifications. Or, the integrator may buy the code and documentation, and then modify each as required. Between these two ends of the spectrum are a range of options to both the program and the integrator.

4.3 SYSTEM DOCUMENTATION STANDARDS AND ANALYSIS

The Data Item Descriptions (DIDs) which have been developed for the series "A Guide to Procurement of Trusted Systems, Volume 3," were written to be applied to products [3]. However, they require the definition of the TCB interfaces and the identification of the TCB protection mechanisms. In the procurement of trusted systems, these DIDs are applicable for system-level assurance documentation. The orientation (e.g., system-level, product-level) of the DID must be expanded outside the framework of the DID. The Statement of Work (SOW) or the Contract Data Requirements List (CDRL) calling out the DID should include statements for the system-level orientation of the resulting assurance documentation. These SOW or CDRL statements should require the examination of the interfaces and mechanisms between products and the analysis of the elimination of interfaces and mechanisms.

A real challenge in the replacement of invalidated product-level documentation is the analysis of the validity of the system-level assurance documentation. The certifier validates the assurance documentation for the system and certifies that the system meets certain requirements. However, it is ultimately left to the accreditor of the system to determine the validity of the assurance documentation for the system and give the permission for the system to operate. There is no other body willing to assess the validity of system-level assurance documentation at this time.

5 CONCLUSION

In conclusion, this paper has presented a single approach to the composition of evaluated products into trusted systems. These evaluated products can be combined into trusted systems with assurance. **The system-level assurances must compensate for any invalidation of the individual products' ratings.** The system-level assurance must document the same types of information that the product-level assurance has documented, i.e. interfaces and mechanisms. The only difference is that excluded and eliminated product mechanisms and interfaces must also be assessed in the system-level documentation. **When procuring these systems, the SOW or CDRL should include direction to the integrator to examine the new interfaces and mechanisms between the products and assess the elimination of interfaces and mechanisms.**

BIBLIOGRAPHY

[1] Department of Defense, "Trusted Computer System Evaluation Criteria" (TCSEC), DoD 5200.28-STD, December 1985.

[2] Modell, Martin E., A Professional's Guide to Systems Analysis, McGraw-Hill Software Engineering Series, McGraw-Hill Book Company, New York, 1988.

[3] NCSC-TG-024, Version 1
Volume 1/4, "A Guide to Procurement of Trusted Systems: An Introduction to Procurement Initiators on Computer Security Requirements," December 1992

Volume 2/4, "A Guide to Procurement of Trusted Systems: Language for RFP Specifications and Statements of Work - An Aid to Procurement Initiators," June 30, 1993

Volume 3/4, "A Guide to Procurement of Trusted Systems: Computer Security Contract Data Requirements List and Data Item Descriptions Tutorial," February 28, 1994

Volume 4/4, "A Guide to Procurement of Trusted Systems: How to Evaluate a Bidder's Proposal Document - An Aid to Procurement Initiators **and** Contractors," (Draft)

[4] Page-Jones, Meilir, The Practical Guide to Structured Systems Design, Yourdon Press, Englewood Cliffs, New Jersey, 1988.

[5] Pressman, Roger S., Software Engineering, A Practitioner's Approach, McGraw-Hill Series in Software Engineering and Technology, McGraw-Hill Book Company, New York, 1987.