A SECURE SDS SOFTWARE LIBRARY

Sara Hadley, Frank Hellwig, Ken Rowe, CDR David Vaurio

National Security Agency
National Computer Security Center
Fort Meade, MD 20755

Abstract

A major challenge facing the Strategic Defense Initiative (SDI) program and the development of the Strategic Defense System (SDS) is the use and distribution of reusable software. The need for reusable software has clearly been established by the ever increasing cost of software versus the cost of hardware. This cost disproportion is magnified in a program with the scope of the SDS. A SDS Software Library will be created in which reusable software can be cataloged, accessed, and distributed. A key attribute of this library is security. The software in the SDS Software Library must be protected from unauthorized access and modification. This paper demonstrates the need for a secure SDS Software Library and the means by which this can be achieved.

Introduction

The Strategic Defense Initiative (SDI) program is an impetus to technology developments on a wide variety of fronts. Two of these fronts are computer hardware and software development. The SDI program is computationally intensive, requiring tomorrow's supercomputers today. This technological pace must be matched by equally intensive software development. The trend over the past decades has shown us that software technology always lags behind hardware technology. Total software costs are rapidly growing as machines become less expensive, and as we uncover more and more problem domains that demand an automated solution [1]. Second and third generation software is hosted on fourth generation machines. Furthermore, when new hardware demands new software solutions, old software is frequently "patched up." More often than not, systems fail in their promise to be extensible and maintainable. In response to this software crisis, the Department of Defense sponsored the development of the Ada programming language and with it, the true potential of reusable software.

The SDI program will make great use of reusable software. There is neither the time nor the money to develop a software system "from scratch" for each new project. In addition to developing new, reusable software, the SDI program will make great use of legacy software, especially in the initial phases of the program. The use of Ada is only a partial solution to the problem of software reuse. No amount of software, whether written in Ada or FORTRAN, will encourage reuse if it is not accessible. For code reuse to be attractive, the overall effort to reuse code must be less than the effort to create new code [2]. Reusable software must be organized in a central repository to which authorized software developers have ready access. It must be organized in such a manner as to provide rapid response to requests for reusable modules to meet the requirements of software developers. Otherwise, software developers will design expensive, single application systems rather than waste time and manpower scouring the countryside for a module that may or may not do the job. This requirement for a repository where reusable software modules are accessible to the SDI software development community has resulted in the Strategic Defense Initiative Organization (SDIO) mandating the establishment of a SDS Software Library at the National Test Facility. [3]

The purpose of this paper is to focus on the need and method of achieving a secure SDS Software Library. This will be addressed in the remaining four sections, each presenting the security issues in succeedingly finer levels of granularity. First, the requirements for a secure library is examined. Second, a conceptual model outlining the required operational capabilities is presented. Third, an implementation is suggested. Finally, the previous material is summarized.

Security

The success of the SDS Software Library can only be assured by the proper application of proven security measures. A failure to do this will result in a library where hostile agents can siphon national secrets without detection. The library will contain a concentrated data base of software revealing a great deal of the capabilities and vulnerabilities of the Strategic Defense System. This concentration of defense software in one location makes the SDS Software Library a high visibility target. The data tables that are included in many classified simulation software packages are a high motivator for illegal penetration into the library. Unauthorized access to the SDS Software Library can result in the compromise of information or corruption of software, thereby resulting in a severe impact to national security.

In a recent tutorial [4], a majority of papers addressed the design and structure of software for reusability. A lesser number were concerned with the actual implementation of a software library. Not a single paper addressed security.

One reason that security is not a principal concern of software library developers is that the principal beneficiaries of reusable software are the software developers themselves. The savings may be passed on to the buyer (i.e., the government), but the actual software development and savings through reusability are an "in house" issue.

If the particular software project is classified, the development is usually accomplished at the system high level where all users are cleared to the highest level of the data. The programmers and the development system are collocated adding to the amount of security which can be enforced through physical means. The situation in the SDS Software Library is quite different due to its handling of material of various classification levels and its distributed user base, therefore making a system high implementation inefficient.

The software stored in the library and the users of the library will span a wide range of classification levels. The flexibility necessary for the library to be responsive to all classification levels eliminates the option of system high operation. Since the SDS Software Library does not actually execute programs, the security treatment is different from that of other computer systems. The true problem is how to enforce security when such a large number of users have access to such a wide range of storage.

Conceptual Model

The SDS Software Library is a storage facility for reusable software serving a widely distributed network of users. The library enables the SDI software development community to efficiently produce complex software systems by providing access to an existing software base. The library is the central point of access to these reusable software modules. A conceptual model of the library is an enumeration of the services and functions the library must provide. The services and functions exist independently of the library's actual physical configuration. It is, however, difficult to create an abstract functional concept without first defining the physical components that constitute a software library. The following will briefly describe the physical elements of the library.

The core of the software library is the computer system, its software, and the associated storage system. The computer system provides on-line library services to the users by executing an application program referred to in this paper as the Software Library Management System (SLMS). The SLMS is the user interface to the information stored in the library and is the system by which the administrative personnel operate the library. The SLMS provides such services as cataloging, tracking, version control, and integrity verification of software items.

It is important to note that the library is not exclusively an electronic storage facility. As in a conventional library, information can be maintained in the form of printed material or microfilm. A user could request the mailing or in the case of extremely sensitive information, the transfer by courier of the requested material. This nonelectronic extension of the library divides the storage media into two groups: on-line and off-line. Material in on-line storage is accessible to the users through the SLMS. Material in off-line storage is transmitted through more traditional means.

The administrative personnel of the library serve two purposes. The first is in the computer system operating staff. The second function is the entry point for software submitted to the library.

So far, we have viewed the library as an information source. Prior to being a source, the library must acquire software items. This acquisition phase is a continuing process. The SLMS is the filter for information flowing from the library to the users. The filter for material being submitted to the library is the administrative personnel. The reasons for a nonautomated mechanism are both technical and philosophical. Current technology does not enable the creation of a system which can examine a piece of code, analyze it, assess its worth in terms of usefulness and reusability, and intelligently catalog it. The second reason involves the human element. The degree of confidence users would have in software obtained from the library is dubious if there is not some intuitive assurance that a human has at least reviewed the software for content and potential.

A conceptual model of the library is derived from a detailed look at the services and functions required of the library to operate smoothly and efficiently. This is formally expressed by a set of required operational capabilities. The required operational capabilities are a functional decomposition of the SDS Software Library. They state what services the library provides to its users and specifies the interaction between the users and the library. The required operational capabilities determine the system level requirements for the library. They are not an all inclusive listing of functions and services. However, the required operational capabilities must be comprehensive so that additional requirements support the composite system without compromising any individual operational capability or degrading one another. The following is a listing of the minimal required operational capabilities.

Fundamental Capabilities

Most computer operating systems enforce a relationship between the users and objects such as files and programs. These relationships are usually expressed as read, write, and execute capabilities. In essence, the SDS Software Library is a system with a large number of users and objects. As a top level specification, the fundamental access capabilities apply as follows:

Read: The library users and administrative personnel will have read access capabilities as specified by their individual access rights.

Write: Only the administrative personnel shall have write access capabilities as specified by their individual access rights.

Append: This is a limited write capability to allow the addition of information to an existing package. The SLMS will account for updates.

Execute: The SDS Software Library shall not have the capability to execute any software item stored in the library. The SDS Software Library is a software repository, not a software development center. The only software the library will execute is the SLMS.

Access Modes

The users of the SDS Software Library shall have two modes of access to the library: system and retrieve.

System: In this mode, the user communicates with the SLMS. A typical function in the system mode is accessing the SDS Software Library Catalog. This mode may however have access limitations (e.g., an unclassified user will not be able to scan classified summaries of classified items).

Retrieve: This mode grants the user direct read access to a software item. Notice that read access is a defacto retrieve mode since it is impossible to control a situation where a user at a remote terminal chooses to download the information appearing on his terminal.

Access Paths

The SDS Software Library shall support the following means through which users may access the library:

Local: Authorized users at the library facility should have direct access to the library.

Remote: Authorized users will be able to access the library via commercial telecommunication links.

Dedicated: The SDS Software Library shall support dedicated electronic communications links. In the case of classified material, these links will be encrypted.

Other: Authorized users may be able to communicate with the library via telephone, mail, courier, and other nonelectronic means.

Access Controls

In accordance with the Department of Defense Trusted Computer System Evaluation Criteria (TCSEC) [5], the library shall incorporate discretionary and mandatory access controls and labels for an A1 system. There will be some differences between the TCSEC and the implementation in the library due to the fact that the library users have no write capability and that the library is not capable of executing stored software items.

Accountability

The SDS Software Library shall incorporate mechanisms to enforce the identification, authentication, and audit requirements as specified for an A1 system by the TCSEC.

Integrity Controls

The SDS Software Library shall maintain software integrity. Integrity mechanisms ensure that the state of a software item is identical (i.e. has not been modified) to its state at a previous time. The proper use of access controls and integrity locks ensure that software is maintained and updated under strict control.

To ensure that data integrity is maintained, the access to that data must be controlled. The SDS Software Library access controls are taken from the TCSEC. David Clark and David Wilson, in an IEEE paper, "A Comparison of Commercial and Military Computer Security Policies," argue that the military security policy only protects information from unauthorized disclosure. If, however, a user is authorized to access a particular data item, there is no restriction on how that data can be manipulated [6]. The protection from unauthorized modification can be implemented in one of two ways. The first is to place integrity protection mechanisms between the user and the information. This protection would be in addition to existing access controls. The second is to eliminate the user's ability to modify information. The latter method is inherent in the read only functionality of the library.

Integrity locks determine if information has been modified. An integrity test can be performed to verify that the present state of a software item is unchanged from some previous state. Additionally, after a software item is distributed to one or more users, the same capability must exist at the remote location to verify that the state of that item matches the state of its parent in the library.

Storage

The SDS Software Library will be required to store software modules having different levels of classification (i.e. multilevel secure storage). Also, certain items may be under strict control independent of classification level. Items under less stringent control should have a wide availability while those under strict control must have very limited distribution.

Since the SDS Software Library is a software storage facility, not a software development center, all storage will appear to be of the write-once-read-many type.

Software Entry

Software items are entered into the SDS Software Library only by the administrative personnel. The submitter of the software item must provide the following:

Header: The header contains the authors, organization date, language, system, and system configuration on which and for which the software item was developed.

Pedigree: A clear and detailed development history of the module. This will specify the current version number as well as previous versions, independent of whether or not those previous version exist in the library. The pedigree shall also state the program for which the item was developed and the test and verification/validation history.

Functional Specification: A textual document detailing the precise function of the software item.

Interface Control Document (ICD): A formal document indicating all entry and exit points, data structures, data types and any other operational requirements necessary to execute the software item per its specifications.

Current technology does not permit the formal verification of software at the code level. The ability to automatically analyze software and determine if it contains trojan horses, viruses, or other malicious functionality is still years away. This leaves no other alternative but to distrust all software entered into the library. The saving grace of this bleak fact is the nonexecutable nature of the library. Programs containing a virus cannot propagate to other programs stored in the library since they will not be executed in the library. It is the user who assumes responsibility for the correct or incorrect functioning of a software item obtained from the library.

Distribution

Software distribution is the transmission of a software item to one or more authorized users via an approved access path. The library shall provide for trusted distribution of software.

Catalog

The library will maintain a catalog of all software items stored in the library. "The library shall contain procedures that help construct queries and evaluate the retrieved sample for potential reusability." [2]

Physical Security

The SDS Software Library central host computer system must be situated in a physically secure area. Protection must be offered to prevent unauthorized access to information and to prevent the malicious destruction of hardware, software, or other library elements in an attempt to deny service.

Implementation

In order for the SDS Software Library to simultaneously process information of different sensitivity levels the Department of Defense requires the library to be a trusted system. The TCSEC defines a trusted system as "A system that employs sufficient hardware and software integrity measures to allow its use for processing simultaneously a range of sensitive or classified information." [5] The SDS Software Library must be designed as a secure system while preserving the required functionality. What will be presented here is an informal implementation outline.

This implementation fulfills the operational requirements through the proper application of information security mechanisms. It will examine only the electronic portion of the library. The nonelectronic areas are addressed by existing policies for the handling of classified material.

At the center of the library is the computer system which controls all of the information between the users and the storage. No user has direct access to the library storage facility. All access is mediated by the central computer system. In addition to enforcing access controls, it incorporates other security relevant functions such as audit trails and user authentication. As stated previously, the fact that the library users and the information stored in the library span all classification levels mandates the central computer to be a trusted system.

The requirements for the type of trusted system are derived from the publication Guidance for Applying the TCSEC in Specific Environments [7]. This standard provides guidance to determine the minimum evaluation class required for a system in specific implementation. The evaluation class determination is based on three factors: the minimum use clearance, the maximum data sensitivity, and the type of system (i.e., open or closed). In the case of the SDS Software Library, the minimum user clearance is unclassified. We will assume the maximum data sensitivity to be top secret. The type of system to be used in the library will be considered a closed system. This means the library will not execute applications software from outside sources. The only application software which the library will actually execute is the SLMS. This will be developed by cleared personnel under a tight, configuration controlled environment. Applying the guidance standard to these factors results in a criteria class requirement of an A1 system.

The SLMS is the application package hosted on the trusted system. The SLMS is the interface through which the users and administrative personnel access and manage the library. Its functions include those library procedures not inherent in the operating system of the trusted computing base. These procedures include the cataloging, tracking, and overall control of the software items stored in the library. The security feature of the SLMS is the integrity locking of all information in the library. There is currently no standard algorithm or device available to perform an integrity lock. When an algorithm or device becomes available, it would be integrated into the software library by either software or hardware.

An A1 system affords the data separation necessary to allow an algorithm to be implemented directly within the SLMS. This procedure could append a cryptographic authentication code to all software items entered into the library. The alternative is a hardware sealing system in line between the computing system and the library storage facility. Any nonsealed item passing from the system to the storage is appended with an authentication code. When an item is transferred from the storage facility it must pass through the hardware system where an integrity test is performed. Any item failing the integrity test would be flagged and rendered unavailable until some corrective action is taken.

The final level of security to consider is the communication channels linking the user to the library. These links must be secure in order to transfer classified information. This requires the use of secure gateways to the library. One possibility is to locate the library within an existing classified data communications system. Sections of the National Test Facility provide this capability. Locating the SDS Software Library within the National Test Facility will provide the SDS Software Library with secure data communication.

Conclusion

In this brief treatment of a complex subject, we have stated the necessity for an SDS Software Library, cited its required operational capabilities, and shown the mandatory role that security must play to create a successful system. A software library is the only method by which the SDI program, or any program of such magnitude, can accomplish its challenging software development tasks. The notion of reusable software demands a central access facility. A software library must provide a broad range of services to entice software developers to make good use of reusable code. None of these services should be degraded unnecessarily by the proper incorporation of security.

The SDS Software Library must be a secure, trusted system to allow the library to handle software of various sensitivity levels. This places stringent requirements on the system as a whole spanning the areas of A1 level trusted mainframes to creating and standardizing secure and verifiable integrity locking mechanisms. There are many challenges to be met in achieving a secure SDS Software Library. Failure to commit to the security of the SDS Software Library will result in unusable, or even malicious, rather than reusable software.

References

[1] G. Booch, Software Engineering with Ada. The Benjamin/Cummings Publishing Company, Inc., 1983, p. 7.

[2] R. Prieto-Diaz, P. Freeman, "Classifying Software for Reusability," IEEE Software, vol. 4, no. 1, pp. 6-16, January 1987.

[3] Strategic Defense Initiative Organization, "Strategic Defense System Software Policy," p. 3, April 7, 1988.

[4] P. Freeman, Tutorial: Software Reusability. Computer Society Press of the IEEE, November 1987.

[5] Department of Defense, Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28 STD, December 1985.

[6] D. D. Clark, D. R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," IEEE Symposium on Security and Privacy, April 1987, pp. 184-194.

[7] Department of Defense, Department of Defense Computer Security Requirements--Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments, CSC STD 003 85, June 1985.