CONF-8906185--1

SAND--89-0737C

DE89 012678

Integrating Security Analysis and Safeguards Software Engineering

Debra D. Spencer, Sandia National Laboratories
Robert M. Axline, Sandia National Laboratories

Introduction

One of the challenges of designing and building a security system to protect critical resources is not only to safeguard the resources being protected, but also to make the security system software itself more secure. The safeguards organization at Sandia National Laboratories (SNL), in conjunction with other Sandia organizations specializing in software security analysis and software engineering, has initiated an effort to address this significant computer security concern in an organized manner. This effort embraces the idea that security must be designed into critical software from the beginning of software development activity in order that later security analysis of the software will provide confidence that the software works only as intended.

Modern safeguards systems protect a variety of resources against adversarial threats. Examples include:

- Storage areas for high-valued assets
- Government administrative and production facilities
- Nuclear power plants
- Aircraft installations

An increasingly common element in these modern safeguards systems is the use of software-controlled devices. Sometimes, software directly controls functions that are critical to preventing unauthorized access.

As modern software systems are developed, they tend to be more and more complex. This complexity provides versatile features that theoretically do a better job of providing the desired functionality. On the other hand, the complexity alone makes the job of assessing the security of the software very difficult.

Adversarial threats to a security system include the usual insiders and outsiders, plus the insider who designed and built the software upon which the security system is based. This special insider has an enormous potential for creating security problems, whether intentionally or not. When no integrity problem exists among the system designers, a knowledgeable outsider may still have the potential for subversion of poorly designed security software.

Many different consequences of security flaws in the system can result. These include:

- Denial of service
- Unauthorized use
- Alteration of critical data
- Leakage of information

Due to 1) the complexity of the software, 2) the opportunities for an adversarial designer, and 3) the subversion potential for poorly designed software, the true level of security for today's security-related software is usually difficult to assess. As a consequence, we believe that the problem of producing secure software is one of great national importance at the present time.

Security in the Software Life Cycle

Opportunities for improving the security of software occur in two primary ways. Software can be developed using tools, techniques, and methodologies which make it inherently more secure. In addition, software can be analyzed during the stages of the development process to determine how secure it is. These two opportunities for improving the security of software are explored in the following two subsections.

Security in Software Design

In recent years, the methods for developing software have changed drastically. Successful software development no longer depends solely on larger and faster computers on which to run it or on innovative computer programmers to create it.

* This work was supported by the United States Department of Energy under Contract Number DE-AC04-76DP00789.

Increasing reliance on computers has driven the demand for software to perform more extensive and increasingly complex tasks. This rapid increase in complexity of software has sometimes overwhelmed those tasked with producing it. For example, inaccurate communication of complex requirements has led to finished software not performing as needed and costing more than expected, and inadequate documentation of the design and implementation has led to increased maintenance costs. The dominating percentage of computer application cost has shifted from hardware in the late fifties and sixties to software development and, especially, maintenance in the eighties.

In response to the problems associated with producing increasingly complex software, the discipline of software engineering has developed. Software engineering is the systematic approach to the specification, development, testing, operation, maintenance, and retirement of software. There are a number of tools, techniques, and methodologies that together form the core of this new discipline. These enhance the quality and productivity of the activities related to software development.

Modern software engineering principles now clearly separate the stages of software development:

- Concept exploration
- Requirements
- Design
- Implementation
- Test
- Installation and checkout
- Operation and maintenance
- Retirement

Specific deliverables are required at each stage of the development in the form of documentation, testing activities, and actual product. This development process provides an excellent structure and has generally been accepted as leading to superior software. However, the problem of building inherently secure software has not been explicitly addressed in software engineering practices.

We believe that some security benefits will accrue to software developed using modern software engineering practices. Examples of these software engineering practices that may lead to enhanced security are:

- Modularity and improved structure
- Improved documentation of requirements and code
- Formal inspection process

We do not believe that these improvements are enough; security-specific measures must be taken in the development process. Effective integration of these security-specific measures into the development process is just beginning. The initiatives we discuss here will aid in this integration.

Software Security Analysis

A security analysis of the software-controlled system is an important step at Sandia National Laboratories in the efforts to assure the security of critical software systems. This process attempts to identify security vulnerabilities, while considering the software in the context of the complete security system. A typical analysis includes assessment of the security attributes of the software and its operating environment as well as static and dynamic analysis of the code. Initiatives presented here should enhance the aspects of structure and automation for the security analysis process.

Security analysts often find themselves in a reactive posture, with the analysis beginning late in the system's life cycle. As a result expensive design modification may be required to make the system adequately secure. One goal of our initiatives is to incorporate more of the philosophy of the security analysis process into the early software design phases. By providing security-related design guidelines and concepts that can be integrated into the earliest stages of the software life cycle, the security of the critical systems can be economically enhanced and these systems can be made easier to analyze.

Software Security Initiatives at SNL

Several groups at Sandia National Laboratories have joined forces to undertake

the mission of designing and developing more secure software. These groups include individuals who design safeguards systems, others who routinely analyze various kinds of software systems for security problems, and still others who are addressing the front-end problem of proper software engineering in the specification and design stages of software creation.

These three groups have well-defined roles in the joint effort. The safeguards software designers are using small software design projects as prototypes for a new methodology of building inherently more secure software. The software engineering and security-analysis groups are developing guidelines for development of software systems that can be more easily analyzed for potential security problems. The security analysts are also developing new analysis methodologies and tools for performing security analysis of software. Together the groups will analyze the success of their efforts from their various unique perspectives.

The goals of the new initiatives are to identify, develop, and promote new tools, techniques, methodologies, and architectures that enhance software and system security and support and improve the software security analysis process. These goals are aimed at eliminating the system's undesired modes of behavior so that it does what it is intended to do and no more. When security problems are present in software despite design methodology aimed at eliminating them, then they should, ideally, be detectable through routine security analysis. Detected security flaws can then be eliminated by incorporating design changes.

The scope of the initiatives is small microprocessor- or microcomputer-based systems, less than or equal to a personal computer in complexity. The focus is on systems that protect critical resources. However, we expect that much of the technology developed for these systems will have broader applicability.

Current safeguards systems that have previously been developed by Sandia tend to be somewhat bigger than the target systems for the initiatives described above. Several of these systems are of the microVAX size. Newer systems under development use personal computers. The decision to restrict the scope of these initiatives to the PC class and smaller makes the problem more manageable.

A system in the class to be addressed by the initiatives is likely to employ a single microprocessor with random-access memory (RAM) for temporary storage of data and read-only memory (ROM) for storage of program code. The system may be embedded, its only interface to the outside world being a cable to which a more complex controller attaches. Another system may have a keyboard and display interface to which an operator has access. In such a case, the software may be either ROM-based or diskette-based. Finally, the system may constitute a minimal network. Examples of systems in the class under consideration are:

- Facility security systems such as access-control portals and intrusion-detection systems
- Material and personnel control and tracking systems
- Storage and retrieval systems for high-valued assets

The approach will be a three-phased one that will build upon past efforts by the groups involved. These past efforts have included:

- A preliminary set of design guidelines for software security
- Preliminary identification of commercially-available software design components with desirable security features
- Promotion of software engineering principles among laboratory software designers
- Informal surveys of some of the current software design tools and software analysis tools
- Development of a concept that uses a dynamic, independent, resident hardware monitor to detect erroneous memory-access behavior in a microprocessor-based system

Phase One objectives are underway and are expected to be delivered before the end of 1990. This activity includes:

- Further development of the software security guidelines
- Design of the resident hardware monitor
- Development of a methodology for integrating security into the early phases of software engineering
- Initial application of the guidelines to a safeguards project
- Investigative work on tools, techniques, platforms, and methodologies for designing and analyzing secure software systems

Phase Two activity is expected to occur during the following two years. This effort will focus on:

- Design and implementation of various custom tools and methodology
- Continued safeguards use of this expanding list of tools and methods
- Assessment of verifiability, cost, and effectiveness of the tools and methodology being developed
- Construction of the resident hardware monitor

Phase Three activity involves long-range projects, whose tasks are more tentative and much less well defined. This activity will build upon accomplishments of the first two phases. Included will be:

- Continued development of tools and design aids
- Promotion of recommended tools, techniques, methodologies, and components
- Quantification techniques for the complexity and security of software systems
- Refinement of both guidelines and analysis methodologies
- Implementation of the resident hardware monitor or other concepts in fielded hardware

Summation

These initiatives will work together to provide more secure safeguards software, as well as other critical systems software. The resulting design tools and methodologies, the evolving guidelines for software security, and the adversary-resistant software components will be applied to the software design at each stage to increase the design's inherent security and to make the design easier to analyze. The resident hardware monitor or other architectural innovations will provide complementary additions to the design to remove some of the burden of security from the software. The security analysis process, supported by new analysis methodologies and tools, will be applied to the software design as it evolves in an attempt to identify and remove vulnerabilities at the earliest possible point in the safeguards system life cycle. The result should be better and more verifiably secure software systems.

VAX is a trademark of the Digital Equipment Corporation.

For further information, the authors may be contacted at:

Sandia National Laboratories
P. O. Box 5800
Albuquerque, New Mexico 87185