Towards a High Assurance Secure Computing Platform

Michael D. DiRossi
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723
michael.dirossi@jhuapl.edu

Abstract

Existing commodity operating systems alone do not satisfy the robust requirements of a general purpose, high assurance, secure computing platform. This paper presents the Trusted Ring Framework, a software architecture that provides a foundation from which to host security services that enhance the security of a commodity operating system.

1 Introduction

An area of increased interest in the information assurance community is the development of general purpose, high assurance secure computing platforms built with commodity software and commodity hardware. Such systems would provide a user-friendly operating environment while simultaneously providing high assurance in the strong enforcement of the platform's security policy. The primary advantages of using commodity software over custom software are that it provides a familiar interface and reduces development, operational, and training costs.

While commodity operating systems (e.g., Microsoft Windows, Linux) provide these benefits, their monolithic designs preclude them from serving as the Trusted Computing Base (TCB) [2] of robust platforms for two primary reasons. First, since a monolithic design does not abide by the principle of least privilege, the compromise of any single functional component within the kernel results in the compromise of every functional component, including the security policy enforcement component. Second, the internal structure of a monolithic kernel is too complex to allow a formal verification of its correctness, and therefore it cannot provide a high degree of assurance in its ability to enforce a security policy.

Intuitively, building a high assurance secure computing platform out of insecure, low assurance components seems paradoxical. However, the goal is not to prove that there are no errors or vulnerabilities in the operating system (OS), but to prove that required security properties of a TCB (e.g., tamper-proof memory) cannot be violated despite the existence of these defects in the commodity OS. The Trusted Ring Framework is a research effort to develop a software framework that can (1) serve as a foundation to host security services that enhance the security of a commodity OS and (2) provide a high assurance base from which to assert that required security properties hold true. Additionally, this research seeks to define a metric for the security afforded by the Trusted Ring Framework, thereby allowing one to determine if the security benefits outweigh the costs of any associated degradation in performance.

2 Research Description

The Trusted Ring Framework is a proof-of-concept software framework that serves as a secure computing base to host a unique class of services that can enhance the security of a commodity OS running on commodity hardware. The Trusted Ring Framework is designed to provide these security services with guaranteed execution time and a protected memory space that is independent of, and isolated from, the OS. With this architecture, should the integrity of the OS be compromised the integrity of the security service hosted inside the Trusted Ring framework is preserved.

The class of security services that are well suited for the Trusted Ring Framework are those that either assume the integrity of the OS has been compromised or require complete isolation from the OS. As an example security service, one could implement a self-healing capability that would detect modifications made to portions of the OS (e.g., kernel code, system call table, etc.) and restore them to their known good state. Alternatively, one could implement a security service that stores an encryption key on behalf of a user application and performs all cryptographic functions inside a protected area that is isolated from the OS.

Trusted Ring has been designed to take advantage of the security mechanisms provided by the Intel IA-32 architecture [4]. The IA-32 architecture defines four privilege lev-
els, or Rings, with the most privileged level being Ring0 and the least privileged level being Ring3. Current commodity operating systems typically partition the software into two privilege levels with user applications executing in Ring3 and the operating system executing in Ring0; the remaining rings are unused.

The Trusted Ring Framework is composed of user applications, a commodity OS, security services, the Trusted Ring microkernel (TR\(\mu\)K), and the communications paths between these components. The architecture of the Trusted Ring Framework is depicted in Figure 1 with possible communications paths between components depicted with numbered arrows. The Trusted Ring Framework executes an unmodified commodity OS in Ring2, at a lower privilege level than its designers intended. User applications remain in Ring3 and the system calls (1), the communication mechanism between user applications and the commodity OS, remain intact.

The security services for the platform are hosted in Ring1 and communicate with TR\(\mu\)K through a custom interface (2). For the example cryptographic security service described earlier, the optional communication path between a user application and a security service (4) is required. A similar communication path between the commodity OS and the security service is also possible (3) if a driver or the security module wanted to take advantage of features afforded by a security service. TR\(\mu\)K executes inside Ring0 where it manages system memory via the IA-32 segmentation (Rings) and paging mechanisms and handles all traps and exceptions that occur in lower privilege levels. TR\(\mu\)K does not enhance the security of the platform in and of itself; this is the role of the security service.

To satisfy the requirement of using commodity operating systems to build general purpose, high assurance secure computing platforms, TR\(\mu\)K must be transparent to the OS to the extent that an unmodified OS can perform its normal functions. However, TR\(\mu\)K must be able to ensure the desired security properties are not violated and therefore TR\(\mu\)K does not need to be transparent to the extent that it is undiscoverable by a compromised OS. This functional transparency requirement is less strict than full virtualization requirements, which state the OS must be presented an exact replica of the underlying hardware at all times [5]. However, functional transparency is more strict than para-virtualization, which modifies the OS to support a performance-optimized interface [3].

The Trusted Ring Framework can complement virtualization technologies by enhancing the security of the OS inside a virtual machine or by enhancing the security of a hypervisor. Additionally, TR\(\mu\)K is simpler in design and smaller in size than current hypervisors, which should greatly simplify formal verification.

A related goal of this research is to measure how much more secure a system is with the Trusted Ring Framework and a given security service than it is without. One avenue of exploration in this direction is to define a set of attack trees that include the vulnerabilities attackers can exploit to compromise an OS. This set of attack trees can then be analyzed against the same OS hosted inside the Trusted Ring Framework with a specific security service.

Currently, a working prototype is being developed to demonstrate the feasibility of the Trusted Ring Framework, including the self-healing security service described earlier. The OS selected for the prototype is a standard installation of Fedora Core 1 Linux operating in runlevel 3. Additionally, initial steps have been taken with formal methods experts at the University of Texas to extend their ACL2 model of the Y86 processor [1] to accurately model the necessary features of the IA-32 platform. This model is an integral part of proving the desired security properties of TR\(\mu\)K.

References