Design and Implementation of an Extended Reference Monitor for Trusted Operating Systems

Hyung Chan Kim¹, Wook Shin², R.S. Ramakrishna¹, and Kouichi Sakurai³

¹ Department of Information and Communications,
  Gwangju Institute of Science and Technology,
  Gwangju 500-712, Rep. of Korea
  {kimhc, rsr}@gist.ac.kr
² Department of Computer Science,
  University of Illinois at Urbana-Champaign,
  IL 61801, USA
  wookshin@uiuc.edu
³ Faculty of Computer Science and Communication Engineering,
  Kyushu University, Fukuoka 812-8581, Japan
  sakurai@csce.kyushu-u.ac.jp

Abstract. Conventional access control schemes have supported confidentiality and integrity acknowledging the necessary organizational security policy in operating systems. However, many runtime attacks in operating systems involve behavioral semantics, indicating that attacks should be seen as a sequence of access operations. Ironically these attacks are legitimate under any access control policy. This is due to the lack of behavioral dimension in security enforcement. We propose an extended reference monitor to include this dimension. Our method is based on safety property specification on system call sequences. The reference monitor checks the trace at runtime for behavior control in Linux operating system.

1 Introduction

A Trusted Operating System (TOS) is an operating system that includes a security kernel providing protection from diverse threats. The security kernel approach assists in realizing the reference monitor with a trusted computing base (TCB) that enforces the security policy of a given system [1]. Most recent TOS projects have incorporated enhanced access controls such as Mandatory Access Control (MAC) and Role-Based Access Control (RBAC) beyond conventional Discretionary Access Control (DAC). MAC enables the TOS to operate with mandated information flow from low to high [2]. RBAC [3] offers flexibility in enforcement and policy neutrality. It also lessens the burden of administration. Both present advantages covering the lack of DAC to enhance OS’s access control service.
As for enhanced access control, researchers seem to commonly adopt privilege transition mechanism in the prototype. GRSecurity’s role transition [17] and SELinux’s domain transition [18] provide examples of the trend. These are analogous to the set-user-id (setuid) mechanism in generic Unix or Linux OS. Usually privilege transition is needed when a user has to perform a special task such as changing the password, configuring the system, or enabling/disabling hardware devices. Security administrators insist on safe execution of such programs, i.e., bounded execution by operations defined (compiled) in the program binary. However, we have constantly witnessed runtime bugs in such programs from security advisories. If the program is subverted by Stack Overflow attack [4], for example, the attacker can hold the transferred privilege of the program; and this is clearly not intended by the administrators. If we carefully observe the attack process, we find that the underlying DAC access control context is valid even though the attack has succeeded. The attacker’s subversion relies on the change of behavior of the program under legitimate access control context. The situation with TOS is no different if it (the TOS) includes the privilege transition mechanism in its access control scheme for whatever reason. This deficiency is due to the lack of another dimension in security enforcement setting aside the access control.

In this paper, we propose an extended reference monitor mechanism for TOS. We were motivated by the aforementioned observation. The security enforcement should be two dimensional. The access control checks the right of the subject to perform a certain operation on an object. Any access control scheme may be adopted in this regard. We extend the enforcement facility with a view to constrain the behavior of operations. This addresses the lack of access control thereby countering program attacks that might succeed due to issues connected with behavioral semantics. Our approach checks the runtime trace based on behavioral specifications. A proof-of-concept implementation in Linux kernel is presented to support our approach.

The rest of this paper is organized as follows. In section 2, we present motivation for the work. We propose the extended enforcement model in section 3. An example policy specification will be given in section 4. Section 5 presents a proof-of-concept implementation of the proposed mechanism. Discussion and related work will be given in section 6. We conclude this paper in section 7.

2 Motivating Examples: Analysis of Program Attack

We encounter various types of program runtime attacks such as Stack and Heap Overflow, Double Free, Format String, besides Race Condition related attacks. Most attacks aim to mess up the privilege beyond the currently active user’s rights. The problem is that the conventional access control in Unix can not effectively counter these attacks. This is because the invasion proceeds under the cover of legitimate access control context. In this section we analyse two types of attack from the point of view of program behavior: Stack Overflow [4] and Time-Of-Check-To-Time-Of-Use (TOCTTOU) [5]. These are representative examples of such attacks involving a single process and concurrency among multiple processes respectively.