Design and Implementation of the TrustedBSD MAC Framework

Robert Watson, Brian Feldman, Adam Migus, Chris Vance
Network Associates Laboratories *
15204 Omega Drive, Suit 300
Rockville, MD 20850
rwatson@nai.com, bfeldman@nai.com,
amigus@nai.com, cvance@nai.com

Abstract

Developing access control extensions for operating systems is an expensive and time-consuming task. Mechanisms available for access control extension lag behind industry standard extension solutions for file systems, process sc

3.3. Stacked File Systems

File systems store persistent data for both the operating system and applications, and as a result are common targets for security research. Security research is just one potential target of file system research requiring extensibility: reliability, namespace transformation and data transformation have all driven the development of stackable file systems. With this model, new services are “layered” over an existing file system by by wrapping operations on file system objects. In a thaB10.679330 Td(de)Tj simil10.44 9 0 Td(on)Tj to.7599 05Td(system)Tj 31.3199 0 Td(on)Tj c 13.43998 0 Tdith

5. Kernel Framework Approach

The TrustedBSD Mandatory Access Control (MAC) Framework is designed to address these problems in kernel security policy augmentation. As FreeBSD is open source, and the FreeBSD Project is willing to accept system extension services, we are able to adopt an approach that assumes that the operating system vendor understands the need for reliable and extensive augmentation for the purposes of security. Our primary design

6.3. MAC Interface to User Processes

The MAC Framework provides user APIs

MAC_PERFORM assumes that a policy entry point has no return value, and is used to post an event to interested poli-

implement per-label storage, reference-counted storage, or statically allocated storage. Kernel objects are pooled by the kernel slab

and creation entry points, the MAC Framework also relies on a per-file system implementation of VOP_SETLABEL, which pushes a label change to the vnode's persistent label store, and then updates the vnode label if successful.

Vnode labels are protected by the vnode reader/write lock, which

MAC Framework, as well as a number of sample policy