CERIAS

the center for education and research in information assurance and security

Usable Mandatory Integrity Protection for Operating System

Ninghui Li, Ziqing Mao and Hong Chen

Motivations

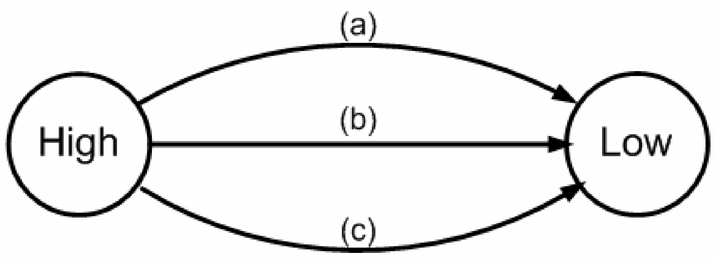
- Host compromise is the root cause of many security problems, e.g., Worms, Botnets, spam, phishing, DDoS
- Having Mandatory Access Control in OS is essential for defending against attacks
  - Softwares are buggy
  - Discretionary access control is insufficient
- Existing MAC systems are difficult to use (e.g., SELinux)

Design Overview

- Objective: Protect host integrity against network-based attacks
- Emphasize on usability: (1) Easy configuration; (2) Compatibility

The UMIP Default Policy

Each process has an integrity level, either high or low
When a process is created, it inherits the parent's IL
The state-transition rules for processes:
(a): receive remote network traffic
(b): receive IPC traffic from a low-integrity process
(c): read a low-integrity file



No restrictions on high-integrity processes.
Restrictions on low-integrity processes:
1. cannot execute any capability
2. cannot write to a file that is not world-writable
3. cannot read a file that is owned by a default account and not world-readable

Implementation & Evaluation

- Implemented using Linux Security Module
- Security: Can prevent most network attacks
- Usability: A small and easily understandable policy
- Performance: 5% overhead on average

Six Design Principles for Usable Access Control

1. ***Provide "good enough" security with a high level of usability, rather than "better" security with a low level of usability***
2. ***Provide policy, not just mechanism***
3. ***Have a well-defined security objective***
4. ***Carefully design ways to support exceptions***
5. ***Rather than trying to achieve "strict lease privilege", aim for "good-enough least privilege"***
6. ***Use familiar abstractions in policy specification interface***

Exceptions in UMIP

- High (RAP): maintain the integrity when (a) happens
- High (LSP): maintain the integrity when (b) happens
- High (FPP): maintain the integrity when (c) happens
- Low (SP): can do operations allowed by special privileges

* Exceptions are enabled when the corresponding binaries are executed

Sample Policy (for ftp server)

- **Program Path:** /usr/sbin/vsftpd
- **File SPs:** (/var/log/xferlog, full), (/etc/vsftpd, full, R), (/etc/shadow, read)
- **Capability SPs:** CAP_SYS_CHROOT, CAP_SETUID, CAP_SETGID, CAP_NET_BIND_SERVICE

Novel Features (compared with Biba and LOMAC)

- Novel concepts to model ***partially trusted*** programs
- A file has two integrity level values:
  - (a). Whether it is protected; determined by the DAC permission
  - (b). Whether it is contaminated; dynamically tracked
- Allows low-integrity files to be upgraded to high-integrity by invoking a specific utility program through a high-integrity channel
- Offers limited confidentiality protection
- Uses DAC information to infer MAC labels for objects, easy to deploy.

PURDUE UNIVERSITY

CERIAS

Discovery Park e-Enterprise Center