Advanced Systems Security: Security-Enhanced Linux

Trent Jaeger
Systems and Internet Infrastructure Security (SIIS) Lab
Computer Science and Engineering Department
Pennsylvania State University

March 2, 2010

Reference Monitor for Linux

- LSM provides a reference monitor interface for Linux
  - Complete Mediation
- You need a module and infrastructure to achieve the other two goals
  - Tamperproofing
  - Verifiability
- *SELinux is a comprehensive reference validation mechanism aiming at reference monitor guarantees*

SELinux History

PENNSTATE 1855

- Origins go back to the Mach microkernel retrofitting projects of the 1980s
  - DTMach (starting in 1992)
  - DTOS (USENIX Security 1995)
  - Flask (USENIX Security 1999)
  - SELinux (2000-…)
- Motivated by the security kernel design philosophy
  - But, practical considerations were made

Inevitability of Failure

PENNSTATE 1855

- Philosophy of the approach
- Flawed Assumption:
  - That security can be provided in application space without proper security features in the operation system (reference monitor)
- Paraphrase: Can't build a secure system without a secure operating system
  - And a secure operating system needs an entire ecosystem
- Come back to this later…

The Rest of the Story

- Tamperproof
  - Protect the kernel
  - Protect the trusted computing base
  - *How to define tamperproofing?*
- Verifiability
  - Code correctness (depends on platform)
  - Policy satisfy a security goal
  - *Not explicitly the focus: Can support MLS for user data*

Design Tamperproofing Policy

- Do not believe that classical integrity is achievable in practice
  - Too many exceptions
  - Commercial systems will not accept constraints of classical integrity
- Instead, focus on providing comprehensive control of access aiming for integrity via *least privilege*
  - Integrity of system components
  - All user processes run with the same label
- *How does least privilege affect access model?*

SELinux Policy Model

- A subject's (process's) access is determine by its:
- User
  - An authenticated identity
  - Are assigned to a set of roles (only one role at a time)
- Roles
  - Identifies a set of types (labels) that a process can attain
- Types
  - The specific subject label for the process now
    - Determines the permissions based on the MPS

SELinux Policy Model

- Subjects and objects have a security context
- For subjects
  - A *context* is a combination of its *user, role, and type*
- For objects
  - A *context* is determined by its type (although placeholders are used for user and role)
- *The accessibility of a subject to an object are dependent upon each's type (label) and authorized ops*
  - Standard MPS protection state

SELinux Policy Rules

- SELinux Rules express an MPS
  - *Protection state*
  - *Labeling state*
  - *Transition state*
- All are defined explicitly
  - Tens of thousands of rules are necessary for a standard Linux distribution
    - Remember, we are ignoring user processes too (other than confining them relative to the system)
- Policy rules: see slide 1-13 in 07-TypeEnforcement

SELinux In Action

- For user to run passwd program
  - Only passwd should have permission to modify */etc/shadow*
- Need permission to execute the passwd program
  - *allow user_t passwd_exec_t:file execute* (user can exec passwd)
  - *allow passwd_t passwd_exec_t:file entrypoint*
- Must transition to passwd_t from user_t
  - *allow user_t passwd_t:process transition* (can run with passwd perms)
  - *type_transition user_t passwd_exec_t:process passwd_t*
- Passwd can the perform the operation
  - *allow passwd_t shadow_t:file {read write}* (can edit passwd file)

Configuring a Program for SELinux

- Goal is *least privilege*
- Function
  - Find the permissions that a program may need
- Configure the policy for these permissions
- Example: *who*
  - See slides 8-13 in 13-Editing…

Take Away

- SELinux: a comprehensive Linux Security Module
  - Aim is to provide a secure OS foundation to commercial systems
- Goal: tamperproofing of system's trusted computing base
  - However, strong integrity guarantees are difficult in a commercial system
  - Aim for least privilege
- Key task is the design of the SELinux policy
  - Complete, but complex ("assembly language of security")