Mandatory Access Control in Linux

CMPSC 443 - Spring 2012
Introduction Computer and Network Security
Professor Jaeger
www.cse.psu.edu/~tjaeger/cse443-s12/
In the early 2000s

• Root and administrator
  – Many programs needed privilege, so they ran with full system permissions

• Consider a network-facing daemon
  – Services requests at a well-known port
  – Low-numbered, so needs root access
  – But, also accessible to adversaries
  – A bad combination...

• What should we do?
Confining Network-Facing Daemons

- Limit permissions of network-facing daemons
  - “Confine” them

- Keep them confined
  - Cannot change their permissions

- How do we do that?
  - Short answer & a long story...
Mandatory Access Control

• System-Defined Policy
  – Fixed Set of Subject and Object Labels
  – Fixed Permission Assignments
  – Fixed Label Assignments: (e.g., file to object label)
  – Fixed Transitions (e.g., setuid)

<table>
<thead>
<tr>
<th></th>
<th>O₁</th>
<th>O₂</th>
<th>O₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>S₂</td>
<td>N</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>S₃</td>
<td>N</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>W</td>
</tr>
</tbody>
</table>
Multi-Level Security is MAC

Access is allowed if

subject clearance level >= object access class and
object categories subset–of subject categories (read down)

Q: What would write-up be?

Hence,

Bob: CONF., {INTEL})
Trent: TS, {CRYPTO, NUC, INTEL})
Alice: (SEC., {CRYTPO, NUC})
DocA: (CONFIDENTIAL, {INTEL})
DocB: (SECRET, {CRYPTO})
DocC: (UNCLASSIFIED, {NUC})
Why MLS Won’t Work

• Lots of information flows that violate MLS
  – For secrecy
  – And integrity

• Have to manage manually
  – No way...

• So, what do we do?
  – LOMAC
  – MIC
  – Others
    • Type Enforcement
MAC in Linux

- In 2001, Linus Torvalds authorized the development of a reference monitor for Linux
  - So, he didn’t have to choose a single security approach
- Linux Security Modules framework was born
  - LSM defines an interface for reference monitoring modules
  - Anybody could build an LSM!
- Introduced in Linux 2.6
  - Version built for FreeBSD
  - Underway for MAC OS X
  - Also, implemented in a variety of user-space programs (X)
- MAC has been in Trusted Solaris for years...
  - But, only one MLS approach (now includes more)
Linux Security Modules Approach

- Reference monitor interface, module, policy
Where Do Hooks Go?

• What property must an authorization hook placement satisfy?
  – Think reference monitor

• How do you know when you have satisfied this property?
  – Not easy
    – Several missing placements were later identified

• Still looking for an automated method to place authorization hooks in legacy code
MAC and Systems

• What is necessary to be a system that enforces MAC policies?
  – Specify: Mandatory Protection System
  – Enforce: Reference Monitor

• Plus, others
  – Management: Policy development tools
  – Services: MAC-aware services
  – Applications: Work with MAC limitations

• What do these systems look like?
  – We’ll examine SELinux
SELinux

- LSM + much more

 SELinux Bootstrap

 SELinux-aware Services

 System Processes

 Linux Kernel

 (1) Load Policy

 (2) Authenticate

 SELinuxfs

 SELinux LSM

 (3) Syscalls
SELinux uses Type Enforcement

- **MAC Policy**
  - Subjects and Objects Labeled
- **Access Matrix Policy**
  - Processes with subject label
  - Can access object of object label
  - If operations in matrix cell allow
- **Focus: Least Privilege**
  - Just permissions necessary

<table>
<thead>
<tr>
<th></th>
<th>O₁</th>
<th>O₂</th>
<th>O₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>S₂</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>S₃</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
SELinux Protection State

• The permissions in an SELinux system are produced by a runtime analysis (same with AppArmor)

• Step 1: Run programs
  – In a controlled (no attacker) environment
  – No enforcement is on

• Step 2: Audit all permissions used

• Step 3: Generate policy file
  – Give the subject label associated with that program
  – All the permissions in the audit file

• Why does this satisfy confidentiality or integrity?
SELinux Labeling State

- Files and users known to the system at boot-time must be associated with their MAC policy labels
  - Map file paths to labels (regular expressions)
  - Map users to labels (by name)
    - These labels are assigned to their initial processes

- How are new files/processes labeled?

- How does “setuid” work?
SELinux Transition State

- Run the privileged `passwd` program
- Simplified view -- takes 4 policy rules to do this
• How many rules are necessary for a Linux distribution?
  – Labeling State - every file and process
  – Protection State - every subject, object, operation
  – Transition State - every process and file transition on access
Configuring Secure Systems

• How do administrators manage MAC systems?
• **Step 1**: Choose an OS distribution
  – Has a MAC policy already
• **Step 2**: Configure a firewall policy
  – Connects MAC processes with network access to network
  – Most processes are given network access
• **Step 3**: Track vulnerabilities
  – Pick your favorite site - CERT, CVE, BugTraq, SANS, ...
• **Step 4**: Run vulnerability scanners on your system
  – See if you are vulnerable
  – If so, remove/update that program or change network
• **NOTE**: Do not change the MAC policy
MAC Upsides

• Security
  – Limits access of root processes
  – Controls network-facing daemons
  – Protects system processes
  – Protects kernel

• Usability
  – Default configuration with OS Distros
  – Mostly enables system to run
  – Does not require any effort for admins

• Bottom line: MAC is here, but in a more limited way than people expected
(Commodity) MAC Myths

• Security
  – MAC protects one of your processes from another
  – MAC protects one of your processes from another user’s processes
  – MAC controls processes use of network
  – MAC ensures that system processes only receive trusted data
  – MAC makes the adversary compromise several processes to access the kernel
  – MAC enforces confidentiality and integrity
Take Away

• In the early part of the last decade adversaries were taking advantage of weak access protections

• MAC was introduced into commodity systems to prevent this

• MAC threat model is network attacks
  – Network-facing daemons

• MAC and code hardening of these daemons have improved the situation
  – but now escalation from untrusted clients through local exploits is common

• Could SELinux prevent Stuxnet?