



# *Mandatory Access Control*



design by t. j. jones

## Topics

- ✓ Overview
- ✓ Practical Objectives
- ✓ B1/Common Criteria
- ✓ MAC, open source
- ✓ LIDS
- ✓ How LIDS looks to the kernel?
- ✓ General Usage Pattern
- ✓ Installation Procedure
- ✓ Recommended LIDS kernel build options
- ✓ LIDS Configuration File
- ✓ LIDS Commands
- ✓ LIDS ACLs
- ✓ Sample LIDS Configuration & Demo
- ✓ LIDS success stories
- ✓ Potential Problems
- ✓ Future Directions
- ✓ Questions



# ***Mandatory Access Control***



- **Practical Objectives**

- a) B1/Common Criteria component mode environment and certification (hard, important mainly for specialized requirements)
- b) MAC - Streamlined, kernel-level interdiction of root privileges and equivalences, compatible with current systems--ie, system hardening (relatively easier)



# *Mandatory Access Control*



- **B1/Common Criteria**
  - a) SELinux (patent encumbered)
  - b) SGI ob1 (incomplete, defunct)
  - c) TrustedBSD (for BSDs, obviously, in development)



# ***Mandatory Access Control***



- **MAC, open source**
  - a) BSD SecureLevel (limited, very stable, BSD only)
  - b) RSBAC (Stable, research oriented, adding features, may move to B1)
  - c) Medusa/DS9 (Incomplete, in development)
  - d) LIDS
  - e) Lomac (defunct)



# *Mandatory Access Control*



## ▪ LIDS

- a. A kernel-mode MAC system, name is misleading as most of LIDS not concerned with IDS
- b. Rule-based system where subjects and objects are filesystem objects, user identity not considered, so LIDS is a filter over Linux/POSIX behavior



# Mandatory Access Control



## ▪ LIDS continued

c. Rules based combine file Read, Write, Append-  
Only flags with flags based on Linux 2.4+  
CAPABILITY system

d. LIDS adds administrative screen above POSIX  
environment--this may be mandatory as with  
BSD SecureLevels, or alternatively password  
controlled (a LIDS-hidden, RIPEMD-160  
password hash is used to protect the LIDS  
state, in that case)



# *Mandatory Access Control*



## ▪ **LIDS Continued**

- e. An additional stage of "sealing the kernel" takes place after boot. Before sealing, only file ACLs are in affect. After boot, CAPABILITY remaining rules are in affect.
- f. Rules compiled to ACL database using userspace tools (lidsadm, lidsconf), and applied from a kenrel-memory database
- g. Rules applied at VFS layer--all Linux filesystems supported (eg, XFS)



# *Mandatory Access Control*



- **How LIDS looks to the kernel?**

Source code side-by-side comparison



# ***Mandatory Access Control***



## ■ **General Usage Pattern**

1. Build and install LIDS kernel
2. Set initial LIDS password and parameters
3. Install LIDS ACLS, typically using a script
4. Boot LIDS kernel
5. As last boot step (rc.local in Redhat), seal the kernel



# *Mandatory Access Control*



## ■ General Usage Pattern **Continued**

6. If LIDS-prohibited changes must be installed on the fly (e.g., firewall rules), session or global disablement of LIDS will be required while the changes are effected

7. If LIDS configurations must be able to be changed on the fly, session or global disablement of LIDS will be required while the new rules are compiled and LIDS configuration is reloaded into the kernel



# ***Mandatory Access Control***



- **Installation Procedure (Standard kernel patch + build):**

1. Build and test Linux 2.4.18 (Linus)
2. Unpack LIDS src
3. chdir to Linux build directory
4. do "patch -p1 < /usr/src/lids-1.1.1r2-2.4.18/lids-1.1.1r2-2.4.18.patch"
5. make [x|menu]config, edit Makefile, install kernel



# *Mandatory Access Control*



## • Installation Procedure (Standard kernel patch + build) **continued:**

6. chdir to LIDS build directory
7. configure; make; make install
8. Set LIDS password
9. Build initial LIDS configuration
10. Boot/test



# ***Mandatory Access Control***



- **Recommended LIDS kernel build options:**
  1. Allow switching LIDS protections
  2. Allow reloading of config file (unless physical reboots are acceptable if ACLS change)



# *Mandatory Access Control*



- **Kernel Build Options- Notes :**

1. You really must set a LIDS password before booting a LIDS kernel. The LIDS tools build scripts automate this step
2. Use "make VIEW=1" when building LIDS tools, to enable display of current ACLs
3. The "configure" script for the LIDS tools (for version 2.4.18) doesn't take account of changes to the EXTRAVERSION flag.



# *Mandatory Access Control*



- **LIDS configuration files**
  1. lids.conf
  2. lids.cap



# *Mandatory Access Control*



- **LIDS Commands**
  - a. lidsadm
  - b. lidsconf



# *Mandatory Access Control*



- **LIDS ACLs – Concepts**

- Rules are:

- a. expressed as a function of over a filename

- b. applied as a function over an inode

- c. a property of a (*process*) context (*process*), that that may be inherited by child processes. Rules may disallow inheritance, or limit it's depth.



# ***Mandatory Access Control***



- **LIDS ACLS – Types**
  - a. System configuration protections (file immutability and file hiding)
  - b. Append-only logfiles
  - c. Process protections and hiding
  - d. Capability restrictions



# *Mandatory Access Control*



## Sample LIDS Configuration and Demonstration



# *Mandatory Access Control*



- **LIDS success stories**
  - a. enGarde secure Linux
  - b. several of our customers



# *Mandatory Access Control*



- **Potential Problems**

- ✓ LIDS ACLs are expressed as functions applied to an inode.
  1. Inodes are specific to a single filesystem. Each filesystem (mount) has its own inode namespace.



# *Mandatory Access Control*



- **Potential Problems Continued:**

2. LIDS rules do not track file name changes—this is correct for control theory, but problematic for some special cases, including the Linux shadow passwd implementation, which creates a new copy of the /etc/shadow file when any change is made. Our preferred solution has been to migrate accounts to LDAP if more than a few non-discretionary logins are required on the system.



# *Mandatory Access Control*



- **Potential Problems** Continued

3. Current sample scripts for installing rules could use improvement.

4. A few LIDS versions have shown memory leaks that caused rule poisoning—this could interact with combined patches, such as Freeswan or a vendor kernel.



# *Mandatory Access Control*



- **Future Directions**

- a. Linux Security Module

- b. Possible enhancements/license changes for RSBAC or SELinux



# *Mandatory Access Control*



Questions?