Linux Security Modules:
General Security Support for the Linux Kernel
by
Chris Wright, Crispin Cowan, Stephen Smalley, James Morris and
Greg Kroah-Hartman

Wesam Lootah
September 2005
Introduction

• Access
  – “The ability to make use of information stored in a computer system.” Saltzer and Schroeder

• Access Control
  – The process by which access requests are granted or denied.
Linux Access Control

• Linux implements discretionary access control

Is this the only type of access control?
The Problem

- There are many different access control models, each with their own advocates.

- Different environments call for different models of access control.

- Many research projects implemented access control models using syscall interposition or kernel patches.

What is the best access control model for Linux?
Linus’s Dellima

SELinux, DTE, MAC, …hmmmm

What is the right solution?
The answer

- What is the solution to every problem in computer science?

One more level of indirection
Before and After

Before LSM

Access control models implemented as Kernel patches

After LSM

Access control models implemented as Loadable Kernel Modules
The LSM project

• “to allow Linux to support a variety of security models, so that security developers don't have to have the ‘my dog's bigger than your dog’ argument, and users can choose the security model that suits their needs.”, Crispin Cowan

Challenges

- LSM needs to reach a balance between kernel developer and security developers requirements. LSM needs to unify the functional needs of as many security projects as possible, while minimizing the impact on the Linux kernel.

  – Truly generic
  – conceptually simple
  – minimally invasive
  – Efficient
  – Support for POSIX capabilities
  – Support the implementation of as many access control models as Loadable Kernel Modules
How was this achieved

• Linux Kernel modified in 5 ways:
  
  – Opaque security fields added to certain kernel data structures
  
  – Security hook function calls inserted at various points with the kernel code
  
  – A generic security system call added
  
  – Function to allow modules to register and unregistered as security modules
  
  – Move capabilities logic into an optional security module
Security Fields

- Enable security modules to associate security information to Kernel objects

- Implemented as void* pointers

- Completely managed by security modules

- What to do about object created before the security module is loaded?
Hooks

• Function calls that can be overridden by security modules to manage security fields and mediate access to Kernel objects.

• Hooks called via function pointers stored in `security->ops` table

• Hooks are primarily “restrictive”
Hooks

- int vfs_mkdir(struct inode *dir, 
  struct dentry *dentry, int mode) 
  
  { 
    int error; 
    down(&dir->i_zombie); 
    error = may_create(dir, dentry); 
    if (error) 
      goto exit_lock; 
    error = -EPERM; 
    if (!dir->i_op || !dir->i_op->mkdir) 
      goto exit_lock; 
    mode &= (S_IRWXUGO|S_ISVTX); 
    error = 
      <!-- security_ops->inode_ops->mkdir(dir, 
        dentry, mode); 
        if (error) 
          goto exit_lock; 
        DQUOT_INIT(dir); 
        lock_kernel(); 
        error = dir->i_op->mkdir(dir, dentry, mode); 
        unlock_kernel(); 
        exit_lock: 
        up(&dir->i_zombie); 
        if (!error) 
          inode_dir_notify(dir, DN_CREATE); 
          <!-- security_ops->inode_ops->post_mkdir(dir, 
            dentry, mode); 
            } 
            return error; 
            }

  Figure 3: The vfs mkdir kernel function with one security hook 
call to mediate access and one security hook call to manage the 
security 
field. The security hooks are marked by<--. 

Figure 1: LSM Hook Architecture
Security System Call

• Allows security modules to implement new calls for security-aware applications

• Implemented as simple multiplexerer

• Three arguments:
  – ID
  – CALL
  – *args

• Arguments are interpreted by the security module
Registering Security modules

- register_security
- unregister_security

- Allows module stacking for policy composition
- Kernel aware of only on module
- Other modules register with the primary module
Capabilities

• POSIX.1e capabilities logic moved into an optional module.

• Capabilities allow partitioning traditional superuser privileges

• Permissive

• Capable interface and task_struct bit vector left as is.
More Hooks

- Task Hooks
- Program Loading Hooks
- IPC Hooks
- Filesystem Hooks
- Network Hooks
- Other Hooks
Performance overhead

• Microbenchmark: LMBench
  – Compare standard Linux Kernel 2.5.15 with Linux Kernel with LSM patch and a default capabilities module
    – Worst case overhead is 5.1%

• Macrobenchmark: Kernel Compilation
  – Worst case 0.3%

• Macrobenchmark: Webstone
  – With Netfilter hooks 5-7%
  – Uni-Processor 16%
  – SMP 21% overhead
Questions?

Question?