How To Write a Linux Security Module That Makes Sense For You

Casey Schaufler
February 2016
Casey Schaufler

- Unix 32 bit port - 1979
- Smack Linux security module
- Security module stacking
Why Would You Write A Security Module?

• We already have terrific security modules

• I can do anything I want with SELinux

• Writing kernel code is hard
Because It’s Your Best Option

- Existing modules are showing their age
- There *are* things you can’t do with SELinux
- Right way to control kernel resources
Restrictive Controls

- Traditional checks are still done
- UID based checks
- Capability checks
- Can’t override a denial
Security Module Don’ts

• Duplicate an existing module

• Depend heavily on user space helpers

• Inflame Al Viro
The Most Important Principle

• Plagiarize! Let no one else's work evade your eyes. Remember why the good Lord made your eyes, so don't shade your eyes, but plagiarize, plagiarize, plagiarize. Only be sure always to call it please "research".
Things You Need To Know About
The components of a Linux Security Module
Hooks

- Security module data management
- Access checks
- Pick and choose as needed
## Hook Return Values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENOMEM</td>
<td>No memory available</td>
</tr>
<tr>
<td>EACCES</td>
<td>Policy denies access</td>
</tr>
<tr>
<td>EPERM</td>
<td>Privilege is required to do this</td>
</tr>
<tr>
<td></td>
<td>- cap_able()</td>
</tr>
<tr>
<td></td>
<td>- CAP_MAC_ACCESS</td>
</tr>
<tr>
<td></td>
<td>- CAP_MAC_ADMIN</td>
</tr>
</tbody>
</table>
Object Based Hooks

• Affiliated with kernel objects
• Access based on attributes attached to the object
• May be difficult for a human to identify
Path Based Hooks

- Associated with pathnames
- May not uniquely identify an object
  - Symlinks
  - Mount points
- Human friendly
Security Blobs

- Hang off kernel data structures
- Managed by the module
- Completely up to the needs of the module
The Blob, the Secid and the Secctx

- Blob contains whatever you like
- Secctx is a string describing it
- Secid is a 32 bit number
  - One per secctx
  - Never exported
  - Volatile
Major Security Module

• Use security blobs
• You only get one
• Called last
Minor Security Module

• Requires no blobs

• Called after:
  • Traditional controls
  • Capabilities

• Called before any major module
Designing Your Security Module

You know, the one that makes sense to you.
What Do You Want To Protect?

• Objects
• Pathnames
• Processes
• Hunks of data
• Resources
What Do You Want To Protect it From?

- Users
  - Malicious
  - Stupid

- Applications
  - Malicious
  - Badly written

- Network access
How Do You Want To Protect It?

- Deny access
- Log the attempt
- Change some attributes
- Something clever
Maintaining Information

- Security Blobs
  - cred->security
  - file->f_security
  - inode->i_security
  - ipcperm->security
  - key->security
  - msg->security
  - sock->sk_security
  - superblob->s_security
  - tun->security
Process Interfaces

Process Attributes
/proc/pid/attr

- security_getprocattr
- security_setprocattr
- Defined in procfs
- Don’t reuse entries
Object Attributes

Information About Things
Traditional Security Attributes

- User and group IDs
- Access modes
- File types
- File Sizes
- Locks
- Filesystem information
- Don’t overload attributes!
Extended Attributes

- Attached by filesystems
- Privilege required to change them
- As big as you like
Pathnames

What’s in a name?
That which we call a rose by any other name would smell as sweet.

- struct path
- Not very convenient
- Not definitive
  - Mount points
  - Symlinks
  - Hard links
Networking

You may not want to go there

Fig. 76. Trådtelefon.
Try netfilter First

- IPv4 and IPv6
- Packet filtering
  - Stateless and statefull
- Address translation
- Port translation
- Extension APIs
Socket Operations

- Checks on many operations
  - Bind, listen, connect
- Packet delivery
- SO_PEERSEC to pass security attributes
UNIX Domain Sockets

- Access to the file system object
- Access to both sockets
- Hooks for connect and send
Internet Domain Sockets

- Only one end of the operation
- Packet header available on receive
- Support for attribute passing using CIPSO
Audit Trail

Adding to the log
Define Your Audit Data

• include/linux/lsm_audit.h

• common_audit_data
  • Under #ifdef in a union

• Your data is up to you
  • Subject
  • Object
  • Operation
Format the Audit Record

- your_log_callback
- audit_log_format
- audit_lo_untrustedstring
- common_lsm_audit
Security Module Interfaces

Why you want your very own pseudo-filesystem
Why Have Security Module Interfaces?

- Load or change access rules
- Read gathered statistics
- Module configuration
- Avoid adding syscalls or ioctls
Mechanics For sysfs

- sysfs_create_mount_point
- register_filesystem
- kern_mount
Security Module
Stacking
Today and In The Future
Stacking Minor Modules

- **module_add_hooks** in `security_init`

- After `capability_add_hooks`

- Before `do_security_initcalls`
Stacking Major Modules - Today

• One at a time

• Boot line
  • security=module

• CONFIG_DEFAULT_SECURITY="module"

• security/Kconfig
Stacking Major Modules – How To Cheat

- There is only one cred->security

- Add your blob to the blob you want to stack with

- Let the other module alloc and free

- Other module stacked first

```c
struct task_security_struct {
    u32 osid;       /* SID ... 
    ...
    u32 sockcreate_sid;  /* fscr ...
    struct task_module module_blob;
};
```
Module Stacking In The Future

- Still under development
- Several blob options
- Representation of secctx
Wrap Up
Get your questions ready
Have A Good Reason

- Do something useful
- It should be something the kernel can and should do
- Follow up with user space support and documentation
Don’t Reinvent The Wheel

• Generic has been done

• It’s the 21st century

• No one liked Bell & LaPadula
  • Or SELinux …
  • Or Smack …
Show Us Something New

• A model for Application Resources has not been done

• Sensor based controls could be fun

• Security doesn’t have to be dull
Thank You