Multiple Concurrent Security Modules? Really?

Casey Schaufler

September 2013
Please Consider As We Go

• Is this a good idea?
• Is this the right approach?
• What would be better?
Motivation

• Security models are changing
• Monolithic modules take too long
  • Driving security into user space
  • Or worse, “drivers”
• We’re doing it anyway with Yama
Design Choices

- All combinations allowed
- All hooks called every time
- Infrastructure replaced
- Modules minimally changed
Module Ordering

• Modules must be compiled in
• Invoked in order configured
  • `CONFIG_DEFAULT_SECURITY=“apparmor,smack,yama”`
• Overridden by boot option
  • `security=“apparmor,smack”`
How it used to work

• Default vector with capabilities module
• New vector has gaps
  • Filled from default vector
  • Each module calls capabilities
• Replace default vector
• Special case Yama stacking

Call hook from vector
Hook calls cap hook
Call Yama hook
The new scheme

- Infrastructure calls capabilities code
- Each hook has a list of security operations
- Registration puts operations on these lists
- Each hook gets called in order
  - No shortcutting
- Success or last error is returned

Do cap check
Call hooks from list
New module data

- list
  - List headers for hook processing
- order
  - This module’s place and slot
- features
  - The special facilities supported

Present
NetLabel
XFRM
secmark
PEERSEC
Security Blobs

• Modules maintain their own
• Infrastructure maintains its own
  • Allocate when necessary
  • Delete when empty
Inside the security modules

- `isp = inode->i_security;`
- `isp = lsm_get_inode(inode, &smack_ops);`
- `cred->security = value;`
- `lsm_set_cred(cred, value, &apparmor_ops);`
Security IDs - secids

- Modules maintain their own
- Infrastructure maintains an array of secids
- Audit uses `struct secids`

<table>
<thead>
<tr>
<th>struct secids</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SELinux secid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Smack secid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Security Information Import and Export

• User visible attributes
• Networking controls
• Backward compatibility
• Complete reporting
Security Context Format

- `<lsmname>`='`<value>`'[<lsmname>=`<value>`]...
- `smack='User'selinux='unconfined_t'apparmor='unconfined'
- No commas
  - Syntactic sugar
- Output when necessary
- Always respected on input

- If y'all don't like it, propose something better
The Present Configuration

- Compatibility for `/proc` interfaces
  - `CONFIG_PRESENT_SECURITY="<lsmname>"`
  - `CONFIG_PRESENT_SECURITY="(all)"`
  - `CONFIG_PRESENT_SECURITY="(first)"`
- Legacy entries in the `attr` directory only
- Use Context Format only if required
New /proc Interfaces

- /proc/.../attr/context
  - The complete context, unaffected by present
- Directory per module
  - attr/apparmor/current
  - attr/apparmor/exec
  - attr/apparmor/prev
  - attr/smack/current
  - attr/selinux/current
  - ...

New securityfs Interfaces

• /sys/kernel/security
• Read only
• lsm
• present
Networking Features

- NetLabel
  - One CIPSO header
- Secmark
  - One 32 bit value
- XFRM
  - Interfaces based on secids
Networking handling

• Identify module by operations
• Explicitly configured
• First available otherwise
• SO_PEERSEC
  • Module explicitly configured
  • Security context format available
casey@schaufler-ca.com
casey.schaufler@intel.com