Stateful access control using LSM

CS547

Thomas Uphill

Why?

- Maintaining state allows for decisions to be made based on runtime conditions.
- State based policy can be more concise
- State based policy can achieve different results than stateless.

Background

UNIX

Files

Permissions

LSM

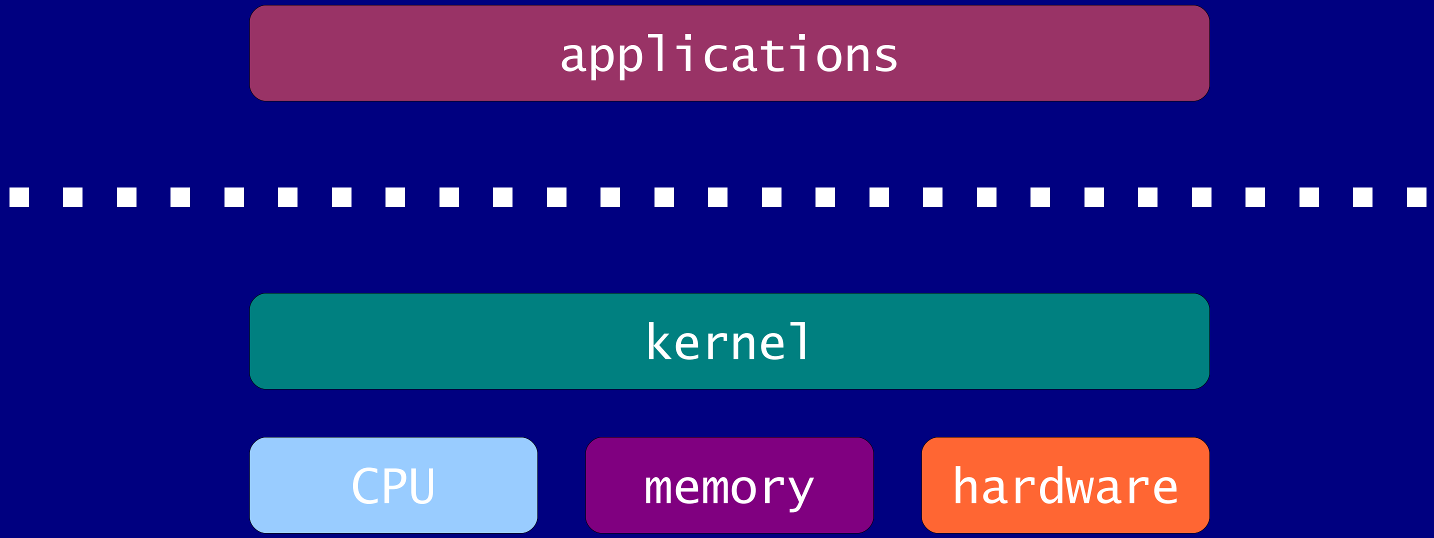
UNIX

- everything is a file
  (keyboards, screens, printers, hardware, kernel internal structures)
- kernel is the master process
  process id (pid) = 0[*]
- pid is unique
  processes have children and parents
- init is pid 1
- /proc filesystem
  contains process information

[*] (some kernel processes appear in as low process numbers,
e.g. [migration/0])

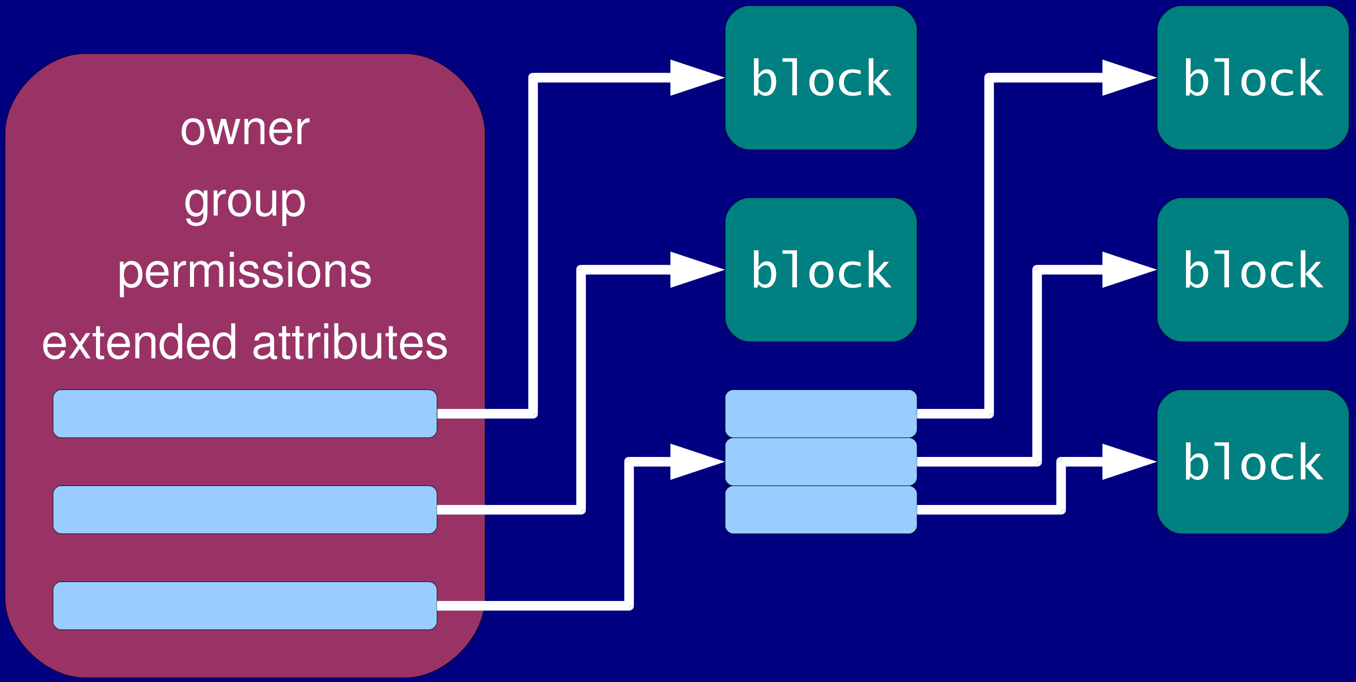
UNIX

- kernel space vs user space



Files

- Files are inodes + blocks
- inodes are information nodes
- blocks contain data on disk

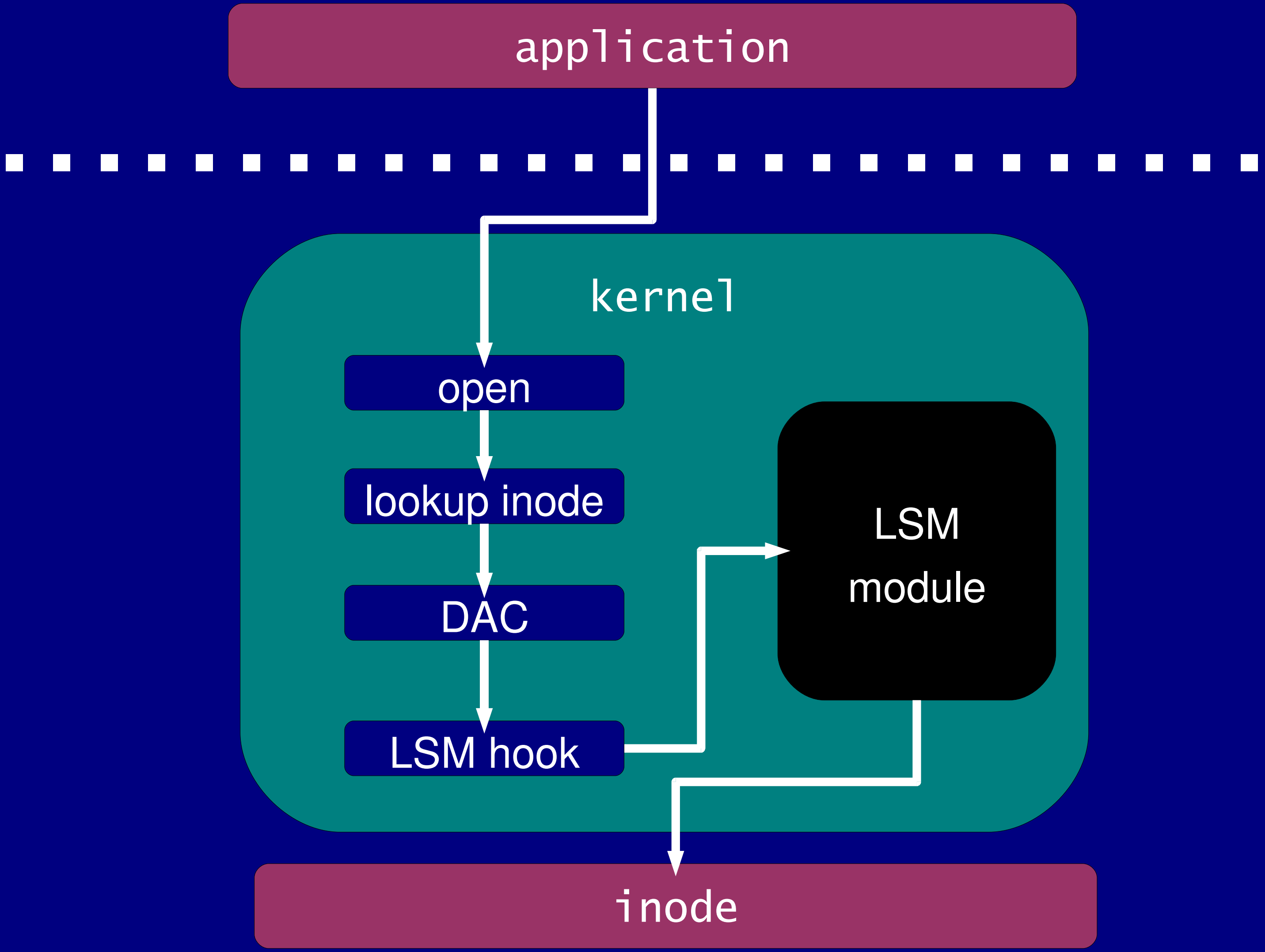


Permissions

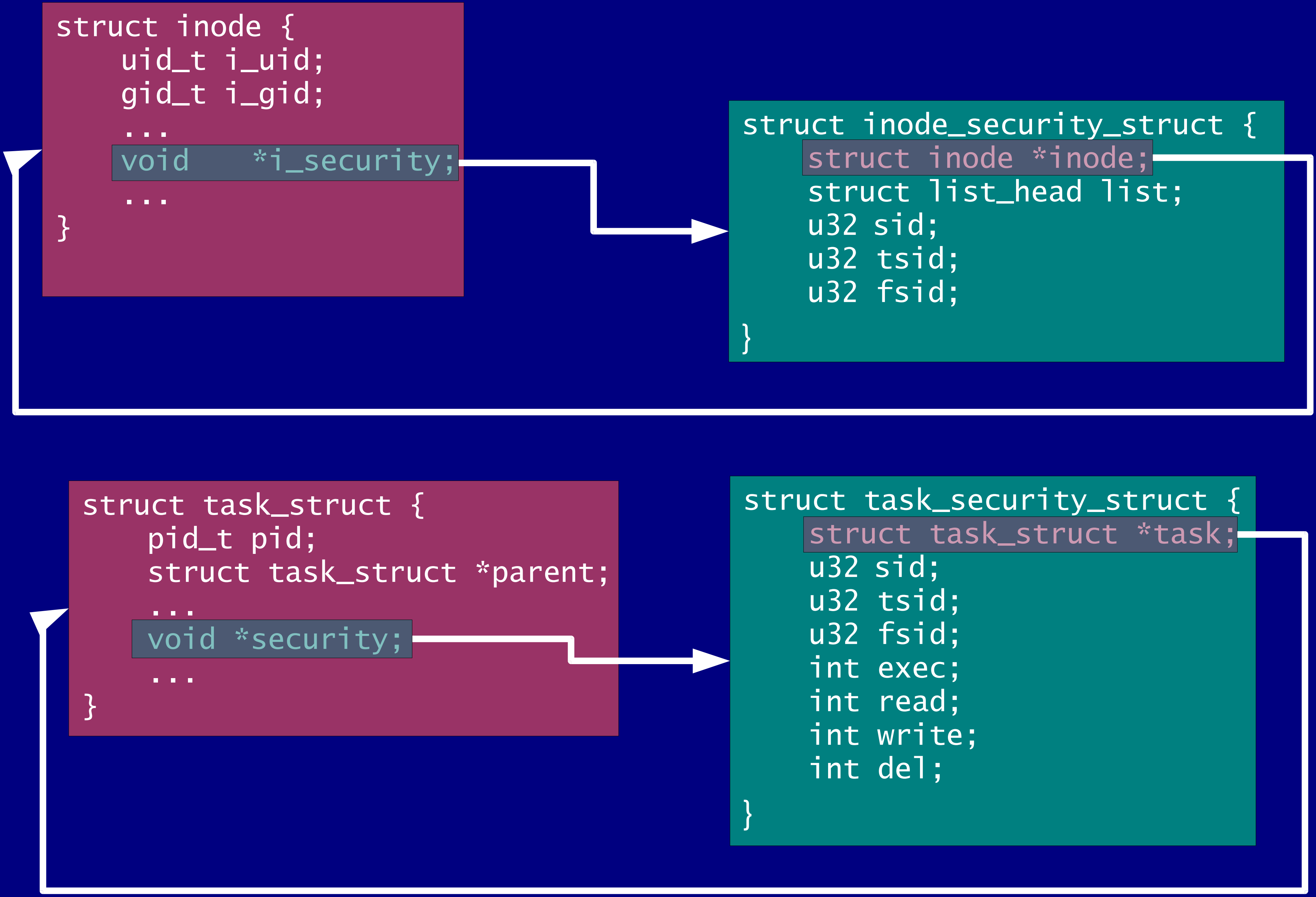
- Classis UNIX permissions:
  user group other
  read write execute
- POSIX Access Control Lists (ACLs):
  list of access control entries (ACEs)
  requires special storage in inodes
  – extended attributes on filesystem
  – access control structure in kernel

LSM

- Linux security module framework
  GNU General Public License
- Crispin Cowan 2001
- hooks
  return 0 to allow
  return non-zero to deny
- security fields
  structs modified



application
kernel
open
lookup inode
DAC
LSM hook
LSM
module
inode



```
struct inode {
    uid_t i_uid;
    gid_t i_gid;
    ...
    void    *i_security;
    ...
}
```

```
struct inode_security_struct {
    struct inode *inode;
    struct list_head list;
    u32 sid;
    u32 tsid;
    u32 fsid;
}
```

```
struct task_struct {
    pid_t pid;
    struct task_struct *parent;
    ...
    void *security;
    ...
}
```

```
struct task_security_struct {
    struct task_struct *task;
    u32 sid;
    u32 tsid;
    u32 fsid;
    int exec;
    int read;
    int write;
    int del;
}
```

Implementation

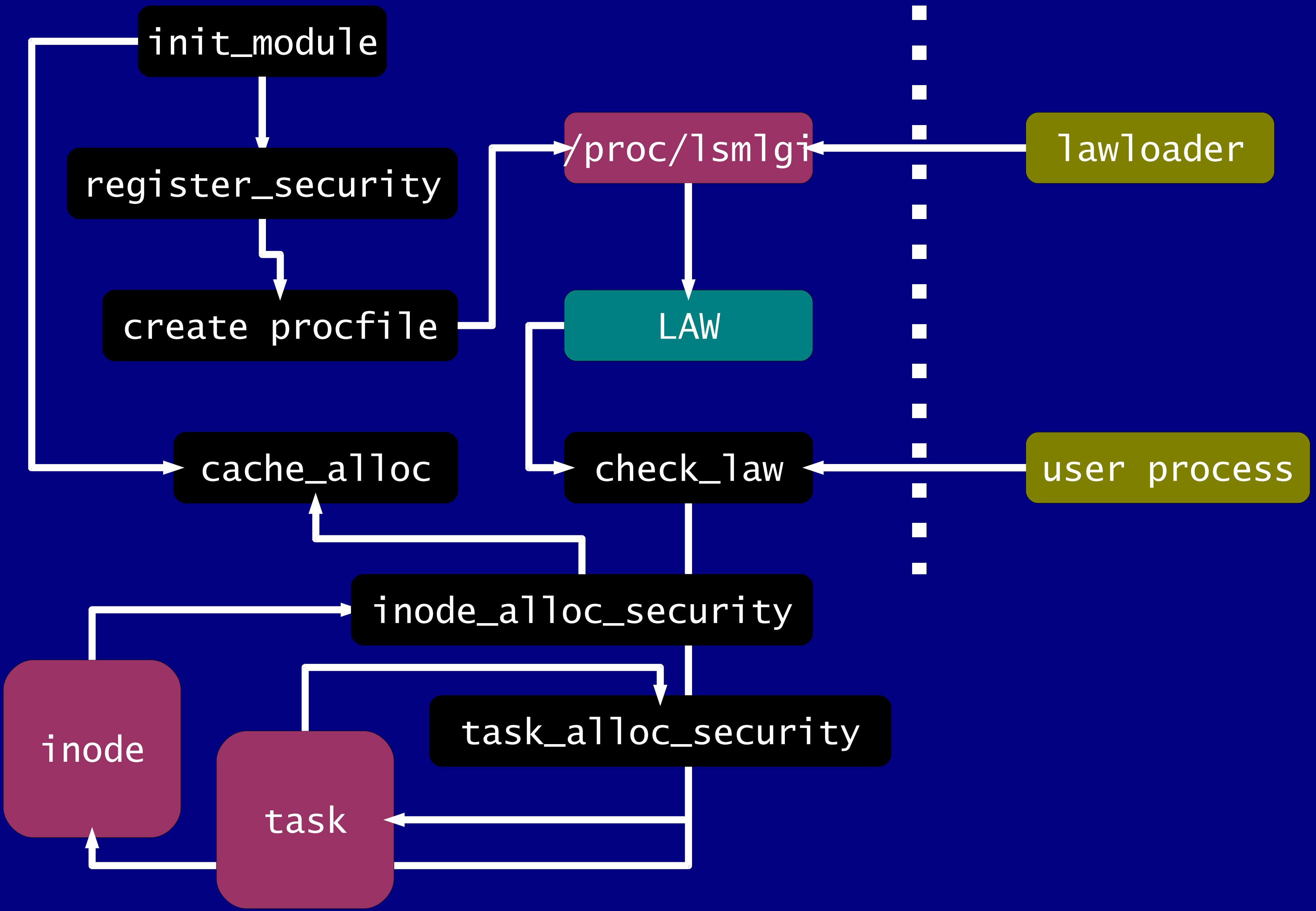
- subset of lsm hooks used
  inode, bprm and task
- inode security cache
  kmem_cache_alloc/kmem_cache_create/kmem_cache_free
- sid /* unique identifier for runtime */
- tsid /* unique identifier for task */
- fsid /* unique identifier for file */
- counters
  read/write/del/exec

Law Language

```
user username operation { action/sid comp action/sid}
group groupname operation { action/sid comp action/sid}
```

Examples:

```
user thomas exec { exec > 20 }
user apache exec { tsid != tsid }
```



init_module

register_security

create procfile

cache_alloc

/proc/lsmlgi

LAW

check_law

lawloader

user process

inode_alloc_security

task_alloc_security

inode

task

Demonstration

visitor.law

Demonstration

apache.law

Demonstration

budget.law

Sources/References

Wikipedia on LSM

http://en.wikipedia.org/wiki/Linux_Security_Modules

LSM Source Code:

http://lsm.bkbits.net

UseNIX Security'02 Abstract:

http://www.usenix.org/event/sec02/wright.html

NSA's SELinux

http://www.nsa.gov/selinux/

Questions/Comments?

http://ramblings.narrabilis.com/wp/linux/stateful-access-control-using-lsm/