ISA 563: Fundamentals of Systems Programming

Authorization and Authentication

Mar. 19, 2012

Authorization and Authentication

- Authentication:
  - Who is the user
  - Is the user who he/she is claiming to be?
- Authorization:
  - Can the user perform operation O on a resource R?

Authentication Methods

- Different ways for authentication:
  - Password database
  - Card readers
  - Biometric information
  - …
- Authentication in programs should be flexible:
  - Should support changes in an authentication method
  - Should support changes in the methods of authentication

PAM: Pluggable Authentication Modules

- Purpose:
  - To separate the development of privilege granting software from the development of secure and appropriate authentication schemes
- How it is done:
  - By providing a library of functions that an application may use to request that a user be authenticated.

(*From Linux−PAM System Administrator's Guide*)

PAM (cont'd)

- Advantages:
  - Easy to write authentication programs
  - Easy to change authentication methods
  - Flexilibity in determining who is allowed to use a service

Demo

pam_auth.c

Authorization

- File system:
  - Read, write, exec permissions for:
    - Owning user
    - Group
    - Others
  - Special permissions:
    - Sticky bit (directories only)
    - Setuid bit
  - Different semantics for files and directories

System APIs

- Changing file permissions

```
#include <sys/stat.h>

int chmod(const char *path, mode_t mode);
int fchmod(int fd, mode_t mode);
```

example 1:

```
chmod("prog", S_IRUSR | S_IXUSR);
```

example 2:

```
int fd = open("prog", O_RDWR);
fchmod(fd, S_IRUSR | S_IXUSR);
```

Demo

modperm.c

System APIs (cont'd)

- Reading Permissions

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int stat(const char *path, struct stat *buf);
int fstat(int fd, struct stat *buf);
int lstat(const char *path, struct stat *buf);
```

Demo

permstat.c

Authorization (cont'd)

- Processes
  - User IDs:
    - Real ID
    - Effective ID
  - Group Ids:
    - Real group id
    - Effective group id

Process IDs

- Every process has two user IDs:
  - Real user ID
  - Effective user ID
- Same for group IDs
- Real UID represents the “original” user id (as returned by getuid() call)
- Kernel only cares about effective UID for most tasks

Changing User IDs

- A process with effective UID of 0 can change its UIDs to any value that it wants
- Any other process can only do one of the following:
  - Set its effective UID to be same as its real UID
  - Set is real UID to be same as its effective UID
  - Swap the two user IDs

System APIs

- Getting information:

```
#include <unistd.h>
#include <sys/types.h>

uid_t getuid(void);
uid_t geteuid(void);
```

- Changing UIDs:

```
#include <sys/types.h>
#include <unistd.h>

int setuid(uid_t uid);
int seteuid(uid_t uid);
int setreuid(uid_t uid);
```

Demo

my_su.c