

2026

CS 332

Subject 13

NJIT

New Jersey's Science &
Technology University

THE EDGE IN KNOWLEDGE

Alex. Gerbessiotis

CS Department



The yellowish/brownish notes are copyright the textbook author.

The bluish notes are copyright the instructor and are add-ons and provide more explanation or additional details



Module 1

*Operating
Systems:
Internals
and
Design
Principles*

Chapter 15 Operating System Security

Ninth Edition
By William Stallings

Authentication

- In most computer security contexts, user authentication is the fundamental building block and the primary line of defense
- RFC 4949 defines user authentication as the process of verifying an identity claimed by or for a system entity
- An authentication process consists of two steps:
 - Identification step
 - Presenting an identifier to the security system
 - Verification step
 - Presenting or generating authentication information that corroborates the binding between the entity and the identifier

NJIT: myUCID

NJIT: used to be called UCID

myUCID credentials:

1. myUCID login name aka UCID

2. myUCID password

Means of Authentication

- Something the individual knows
 - Examples include a password, a personal identification number (PIN), or answers to a prearranged set of questions
- Something the individual possesses
 - Examples include electronic keycards, smart cards, and physical keys
 - Referred to as a *token*
- Something the individual is (static biometrics)
 - Examples include recognition by fingerprint, retina, and face
- Something the individual does (dynamic biometrics)
 - Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm

kinit to obtain a Kerberos ticket
aklog to obtain an AFS token

AFS : uses 12hour tokens (but 10h @ NJIT)

Access Control

Discussed in FileSystems

- Implements a security policy that specifies who or what may have access to each specific system resource and the type of access that is permitted in each instance
- Mediates between a user and system resources, such as applications, operating systems, firewalls, routers, files, and databases
- A security administrator maintains an authorization database that specifies what type of access to which resources is allowed for this user
 - The access control function consults this database to determine whether to grant access
- An auditing function monitors and keeps a record of user accesses to system resources

Compile-time Techniques

■ Choice of programming language

- One possibility is to write the program using a modern high-level programming language that has a strong notion of variable type and what constitutes permissible operations on them
- The flexibility and safety provided by these languages does come at a cost in resource use, both at compile time and also in additional code that must execute at runtime

■ Safe coding techniques

- Programmers need to inspect the code and rewrite any unsafe coding constructs
- An example is the OpenBSD project which produces a free, multiplatform 4.4BSD-based UNIX-like operating system
- Among other technology changes, programmers have under-taken an extensive audit of the existing code base, including the operating system, standard libraries, and common utilities

■ Language extensions and use of safe libraries

- There have been a number of proposals to augment compilers to automatically insert range checks on pointer references
- Libsafe is an example that implements the standard semantics but includes additional checks to ensure that the copy operations do not extend beyond the local variable space in the stack frame

■ Stack protection mechanisms

- An effective method for protecting programs against classic stack overflow attacks is to instrument the function entry and exit code to set up and then check its stack frame for any evidence of corruption
- Stackguard, one of the best-known protection mechanisms, is a GNU Compile Collection (GCC) compiler extension that inserts additional function entry and exit code

Runtime Techniques

■ Executable address space protection

- A possible defense is to block the execution of code on the stack, on the assumption that executable code should only be found elsewhere in the processes address space
- Extensions have been made available to Linux, BSD, and other UNIX-style systems to support the addition of the no-execute bit

■ Address space randomization

- A runtime technique that can be used to thwart attacks involves manipulation of the location of key data structures in the address space of a process
- Moving the stack memory region around by a megabyte or so has minimal impact on most programs but makes predicting the targeted buffer's address almost impossible
- Another technique is to use a security extension that randomizes the order of loading standard libraries by a program and their virtual memory address locations

■ Guard pages

- Caps are placed between the ranges of addresses used for each of the components of the address space
- These gaps, or guard pages, are flagged in the MMU as illegal addresses and any attempt to access them results in the process being aborted
- A further extension places guard pages between stack frames or between different allocations on the heap
- Also used to extend stack space if 'illegal access' is detected (Windows)

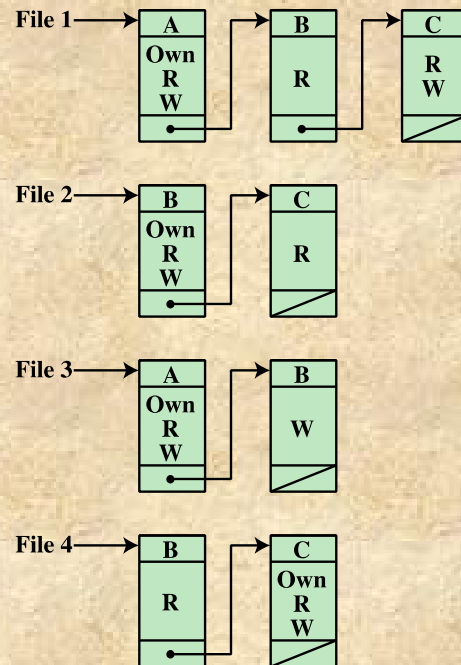
File System Access Control

- Identifies a user to the system **Discussed in FileSystems**
- Associated with each user there can be a profile that specifies permissible operations and file accesses
- The operating system can then enforce rules based on the user profile
- The database management system, however, must control access to specific records or even portions of records
- The database management system decision for access depends not only on the user's identity but also on the specific parts of the data being accessed and even on the information already divulged to the user

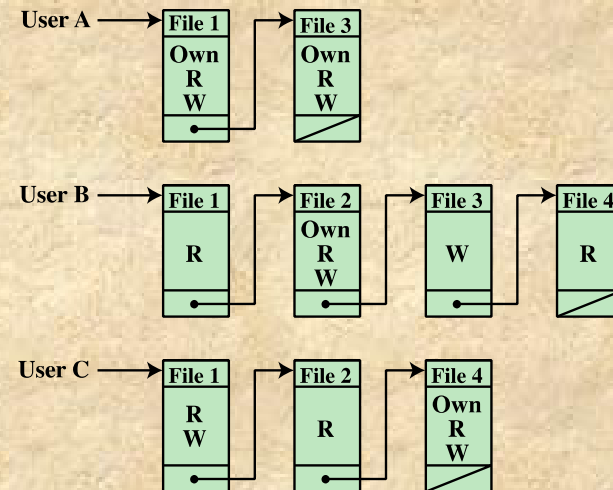
	File 1	File 2	File 3	File 4	Account 1	Account 2
User A	Own R W		Own R W		Inquiry Credit	
User B	R	Own R W	W	R	Inquiry Debit	Inquiry Credit
User C	R W	R		Own R W		Inquiry Debit

Discussed in FileSystems

(a) Access matrix



(b) Access control lists for files of part (a)



(c) Capability lists for files of part (a)

Figure 15.3 Example of Access Control Structures

Access Control Policies

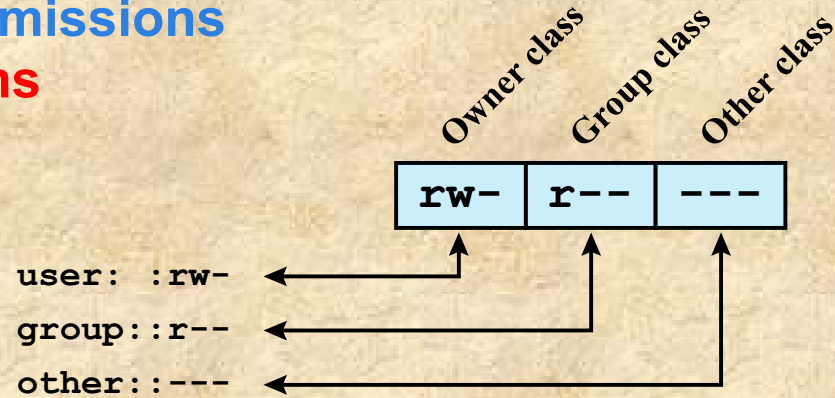
- An access control policy dictates what types of access are permitted, under what circumstances, and by whom
- Access control policies are generally grouped into the following categories:
 - Discretionary access control (DAC)
 - Controls access based on the identity of the requestor and on access rules stating what requestors are allowed to do
 - Mandatory access control (MAC)
 - Controls access based on comparing security labels with security clearances
 - Role-based access control (RBAC)
 - Controls access based on the roles that users have within the system, and on rules stating what accesses are allowed to users in given roles
 - Attribute-based access control (ABAC)
 - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions

Access Control Policies

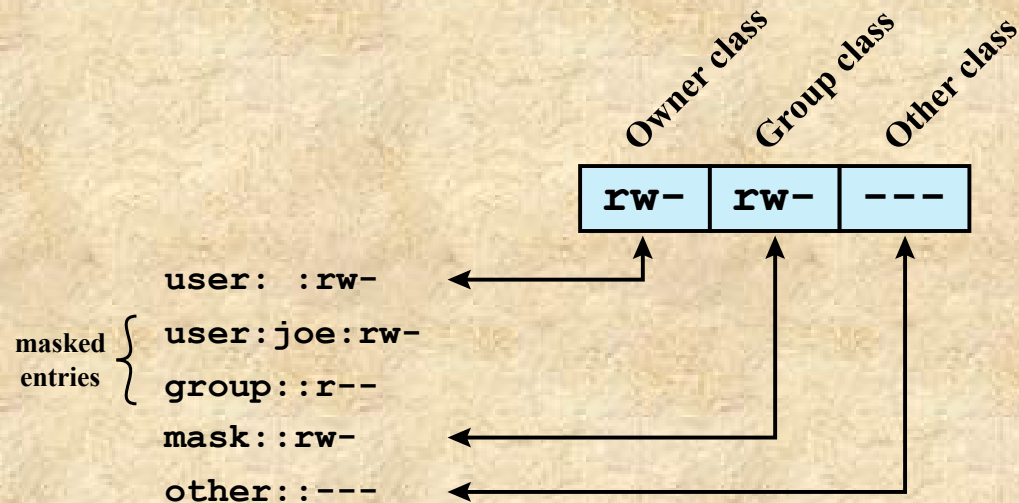
- An access control policy dictates what types of access are permitted, under what circumstances, and by whom
- Access control policies are generally grouped into the following categories:
 - Discretionary access control (DAC)
 - Controls access based on the identity of the requestor and on access rules stating what requestors are allowed to do
 - Mandatory access control (MAC)
 - Controls access based on comparing security labels with security clearances
 - Role-based access control (RBAC)
 - Controls access based on the roles that users have within the system, and on rules stating what accesses are allowed to users in given roles
 - Attribute-based access control (ABAC)
 - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions

Unix/Linux standard permissions

Discussed in FileSystems



(a) Traditional UNIX approach (minimal access control list)



(b) Extended access control list

Figure 15.8 UNIX File Access Control



Unix/Linux standard permissions
can be extended
according to the POSIX standard
for Access Control Lists

getfacl FILE-or-DIR #get ACL of

getfacl -h #help

setfacl FILE-or-DIR #set ACL of

setfacl -h #help

getfacl DIR1 | setfacl --set-file=- DIR2

copy ACL of DIR1 to become ACL of DIR2



```
alexg@alexg-fy19 ~/Z
$ ls
aa  bb  cc  D1  D2

alexg@alexg-fy19 ~/Z
$ ls -l
total 0
-rw-r-xrw-  1 alexg Domain Users 0 Mar 29 23:27 aa
-rwSr-Sr-T  1 alexg Domain Users 0 Mar 29 23:27 bb
-rw-r--r--  1 alexg Domain Users 0 Mar 29 23:27 cc
drwxr-xr-x+ 1 alexg Domain Users 0 Mar 30 00:09 D1
dr-xrw-r--+ 1 alexg Domain Users 0 Mar 30 00:00 D2

alexg@alexg-fy19 ~/Z
$
```

Sticky bit (T not t)
and
SetUID (S not s)
SetGID set (S not s)

POSIX ACL :getfac1 example



```
~/Z  
alexg@alexg-fy19 ~/Z  
$ getfac1 aa bb  
# file: aa  
# owner: alexg  
# group: Domain Users  
user::rw-  
group::r-x  
other::rw-  
  
# file: bb  
# owner: alexg  
# group: Domain Users  
# flags: sst  
user::rw-  
group::r--  
other::r--  
  
alexg@alexg-fy19 ~/Z  
$
```

POSIX ACL :getfac1 example



```
~/Z
$ getfac1 D1 D2
# file: D1
# owner: alexg
# group: Domain Users
user::rwx
group::r-x
other::r-x
default:user::rwx
default:group::r-x
default:other::r-x

# file: D2
# owner: alexg
# group: Domain Users
user::r-x
group::rw-
other::r--
default:user::rwx
default:group::r-x
default:other::r-x

alexg@alexg-fy19 ~/Z
$

alexg@alexg-fy19 ~/Z
$

alexg@alexg-fy19 ~/Z
$
```

Different
permissions
for D1 and D2

POSIX ACL :setfac1 example



```
~/Z
$
alexg@alexg-fy19 ~/Z
$ getfac1 D2 | setfac1 --set-file=- D1
alexg@alexg-fy19 ~/Z
$ getfac1 D1 D2
# file: D1
# owner: alexg
# group: Domain Users
user::r-x
group::rw-
other::r--
default:user::rwx
default:group::r-x
default:other::r-x

# file: D2
# owner: alexg
# group: Domain Users
user::r-x
group::rw-
other::r--
default:user::rwx
default:group::r-x
default:other::r-x

alexg@alexg-fy19 ~/Z
$ |
```

Same permissions
for D1 and D2



AFS (Andrew File System)
extends **ACL** to all files maintained by
it

fs listacl

fs la **#get ACL of ...**

fs setacl

fs sa **#set ACL of ...**

fs copyacl

fs cleanacl

AFS : fs la example



```
alexg@afsconnect2:~  
% ls -l Test/  
total 0  
% ls -ld Test/  
drwx----- 2 alexg users 2048 Mar 30 00:37 Test/  
% fs la Test  
Access list for Test is  
Normal rights:  
  arcsnewstaff rlidwk  
  homer 1  
  system:administrators rlidwka  
  alexg rlidwka  
  http 1  
  httpsvc.webhost03.ucsf.edu 1  
%
```

AFS : fs sa example



```
alexg@afsconnect2:~  
% ls -ld Test  
drwx----- 2 alexg users 2048 Mar 30 00:37 Test  
% fs la Test  
Access list for Test is  
Normal rights:  
  arcsnewstaff rlidwk  
  homer l  
  system:administrators rlidwka  
  alexg rlidwka  
  http l  
  httpsvc.webhost03.ucsf.edu l  
% fs sa Test homer none  
% fs la Test  
Access list for Test is  
Normal rights:  
  arcsnewstaff rlidwk  
  system:administrators rlidwka  
  alexg rlidwka  
  http l  
  httpsvc.webhost03.ucsf.edu l  
% |
```

NJIT

A white curved line that starts under the 'N', arches over the 'J' and 'I', and ends under the 'T'.

New Jersey's Science &
Technology University

THE EDGE IN KNOWLEDGE

Alex. Gerbessiotis

CS Department