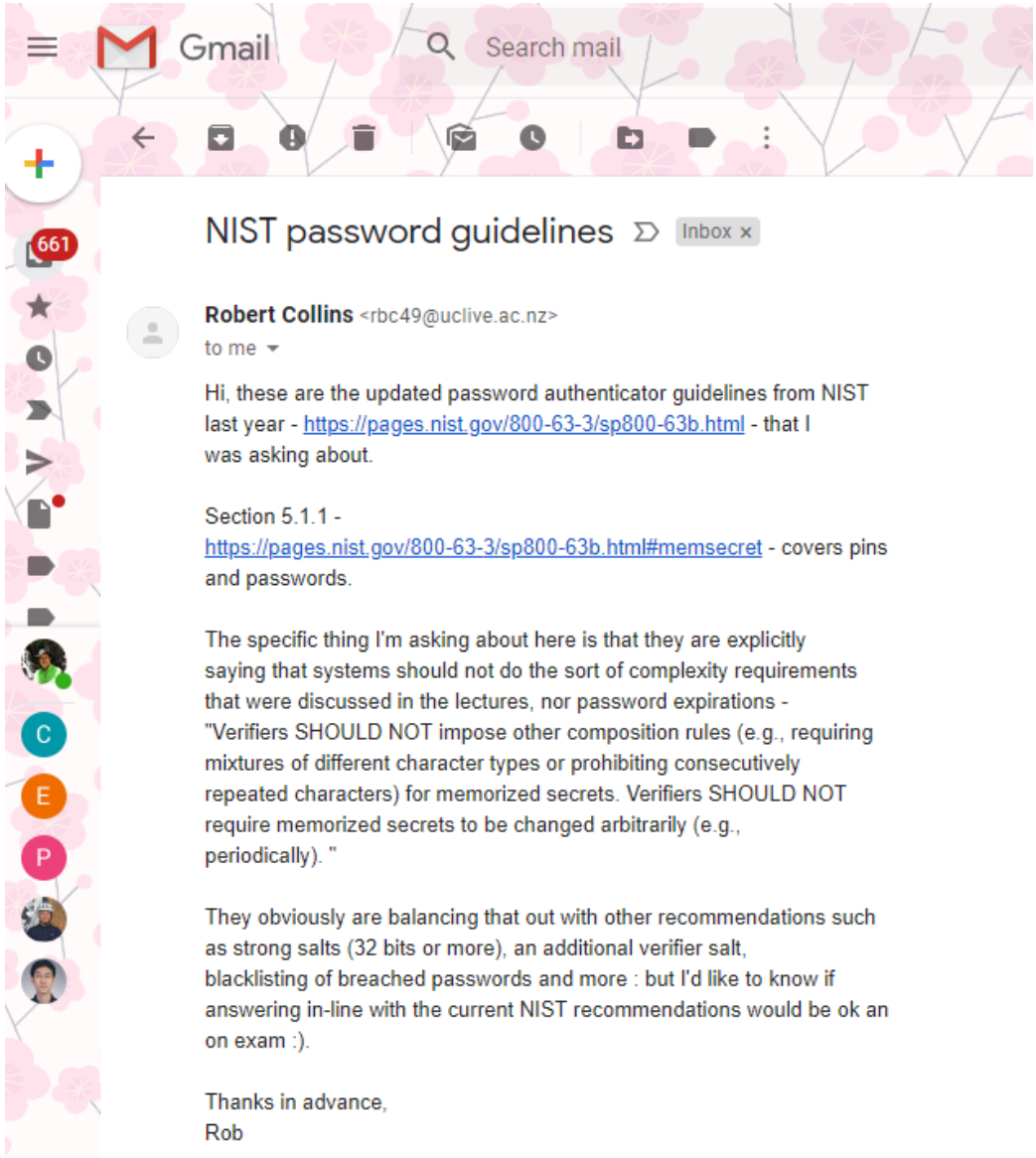
“Extra” (?)


Gmail
Search mail
661
NIST password guidelines
Inbox ×
Robert Collins <rbc49@uclive.ac.nz>
to me
Hi, these are the updated password authenticator guidelines from NIST last year - https://pages.nist.gov/800-63-3/sp800-63b.html - that I was asking about.
Section 5.1.1 - https://pages.nist.gov/800-63-3/sp800-63b.html#memsecret - covers pins and passwords.
The specific thing I'm asking about here is that they are explicitly saying that systems should not do the sort of complexity requirements that were discussed in the lectures, nor password expirations - "Verifiers SHOULD NOT impose other composition rules (e.g., requiring mixtures of different character types or prohibiting consecutively repeated characters) for memorized secrets. Verifiers SHOULD NOT require memorized secrets to be changed arbitrarily (e.g., periodically). "
They obviously are balancing that out with other recommendations such as strong salts (32 bits or more), an additional verifier salt, blacklisting of breached passwords and more : but I'd like to know if answering in-line with the current NIST recommendations would be ok an on exam :).
Thanks in advance,
Rob

Secure | https://pages.nist.gov/800-63-3/

Apps uc sms

National Institute of Standards and Technology
U.S. Department of Commerce

NIST Website | About NIST | usnistgov on Github

Digital Identity Guidelines: Now Available

June 22, 2017

The finalized four-volume SP 800-63 *Digital Identity Guidelines* document suite is now available, both in PDF format and online.

The Trusted Identities Group (TIG) thanks all that contributed to the development of these documents.

PDF versions of the documents are available from:

Document	Title	URL
SP 800-63-3	Digital Identity Guidelines	https://doi.org/10.6028/NIST.SP.800-63-3
SP 800-63A	Enrollment and Identity Proofing	https://doi.org/10.6028/NIST.SP.800-63a
SP 800-63B	Authentication and Lifecycle Management	https://doi.org/10.6028/NIST.SP.800-63b
SP 800-63C	Federation and Assertions	https://doi.org/10.6028/NIST.SP.800-63c

Links to the online version of the SP 800-63 suite are below.




Extra (2)

From: 高可
Sent: Tuesday, 14 August 2018 21:53
To: adr39@uclive.ac.nz
Subject: 高可 sent you an invitation on LinkedIn

LinkedIn

高可 wants to add you to their network

高可

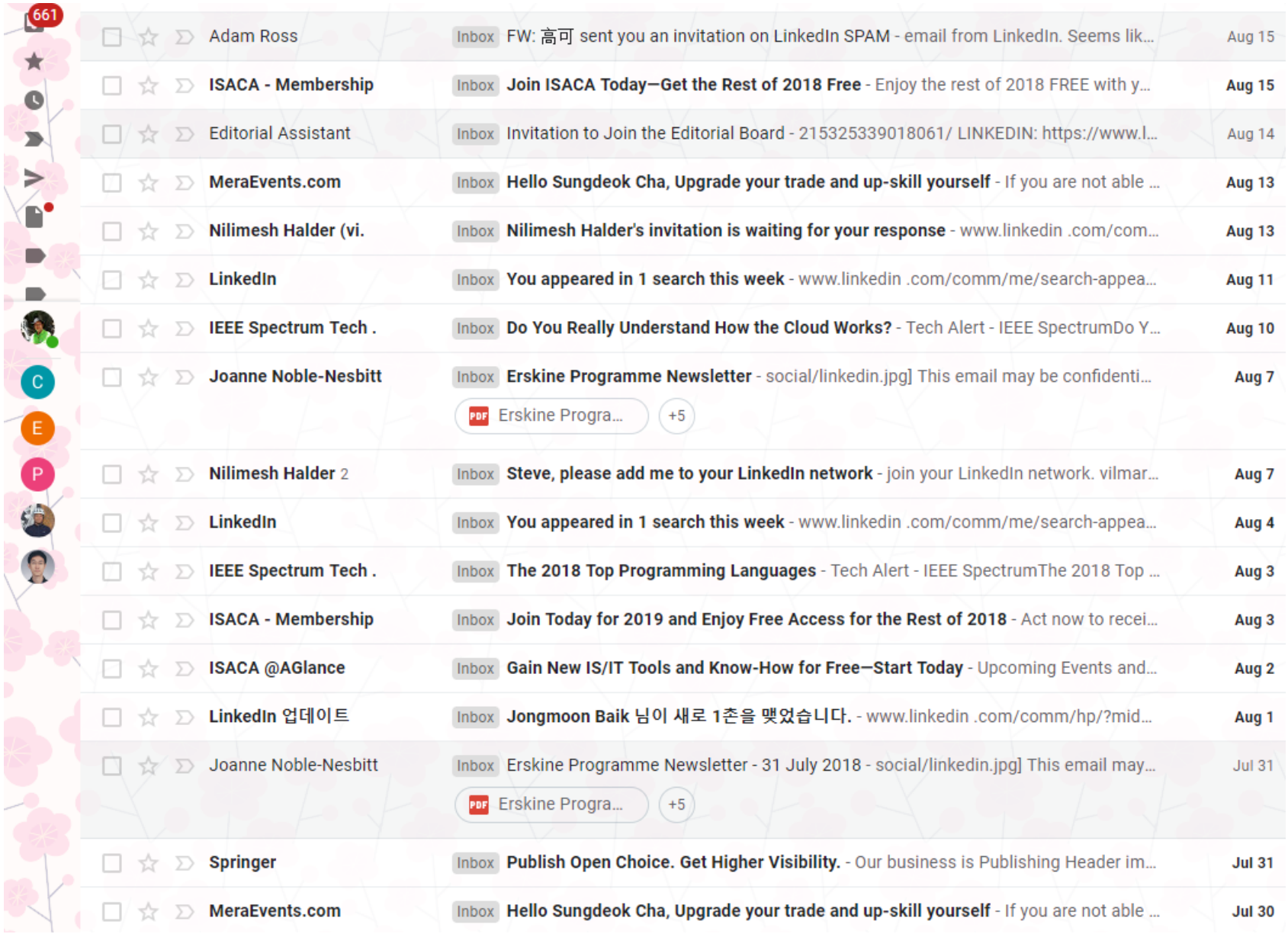
新西兰坎特伯雷大学学生

Canterbury & West Coast, New Zealand · 0 connections

Accept 高可's invitation

LinkedIn is a social network and online platform for professionals. Learn More

Unsubscribe | Help


661
C
E
P
Adam Ross Inbox FW: 高可 sent you an invitation on LinkedIn SPAM - email from LinkedIn. Seems lik... Aug 15
ISACA - Membership Inbox Join ISACA Today—Get the Rest of 2018 Free - Enjoy the rest of 2018 FREE with y... Aug 15
Editorial Assistant Inbox Invitation to Join the Editorial Board - 215325339018061/ LINKEDIN: https://www.l... Aug 14
MeraEvents.com Inbox Hello Sungdeok Cha, Upgrade your trade and up-skill yourself - If you are not able ... Aug 13
Nilimesh Halder (vi. Inbox Nilimesh Halder's invitation is waiting for your response - www.linkedin .com/com... Aug 13
LinkedIn Inbox You appeared in 1 search this week - www.linkedin .com/comm/me/search-appea... Aug 11
IEEE Spectrum Tech . Inbox Do You Really Understand How the Cloud Works? - Tech Alert - IEEE SpectrumDo Y... Aug 10
Joanne Noble-Nesbitt Inbox Erskine Programme Newsletter - social/linkedin.jpg] This email may be confidenti... Aug 7
Erskine Progra... +5
Nilimesh Halder 2 Inbox Steve, please add me to your LinkedIn network - join your LinkedIn network. vilmar... Aug 7
LinkedIn Inbox You appeared in 1 search this week - www.linkedin .com/comm/me/search-appea... Aug 4
IEEE Spectrum Tech . Inbox The 2018 Top Programming Languages - Tech Alert - IEEE SpectrumThe 2018 Top ... Aug 3
ISACA - Membership Inbox Join Today for 2019 and Enjoy Free Access for the Rest of 2018 - Act now to recei... Aug 3
ISACA @AGlance Inbox Gain New IS/IT Tools and Know-How for Free—Start Today - Upcoming Events and... Aug 2
LinkedIn 업데이트 Inbox Jongmoon Baik 님이 새로 1촌을 맺었습니다. - www.linkedin .com/comm/hp/?mid... Aug 1
Joanne Noble-Nesbitt Inbox Erskine Programme Newsletter - 31 July 2018 - social/linkedin.jpg] This email may... Jul 31
Erskine Progra... +5
Springer Inbox Publish Open Choice. Get Higher Visibility. - Our business is Publishing Header im... Jul 31
MeraEvents.com Inbox Hello Sungdeok Cha, Upgrade your trade and up-skill yourself - If you are not able ... Jul 30

CNN tech BUSINESS CULTURE GADGETS FUTURE STARTUPS CNNMONEY

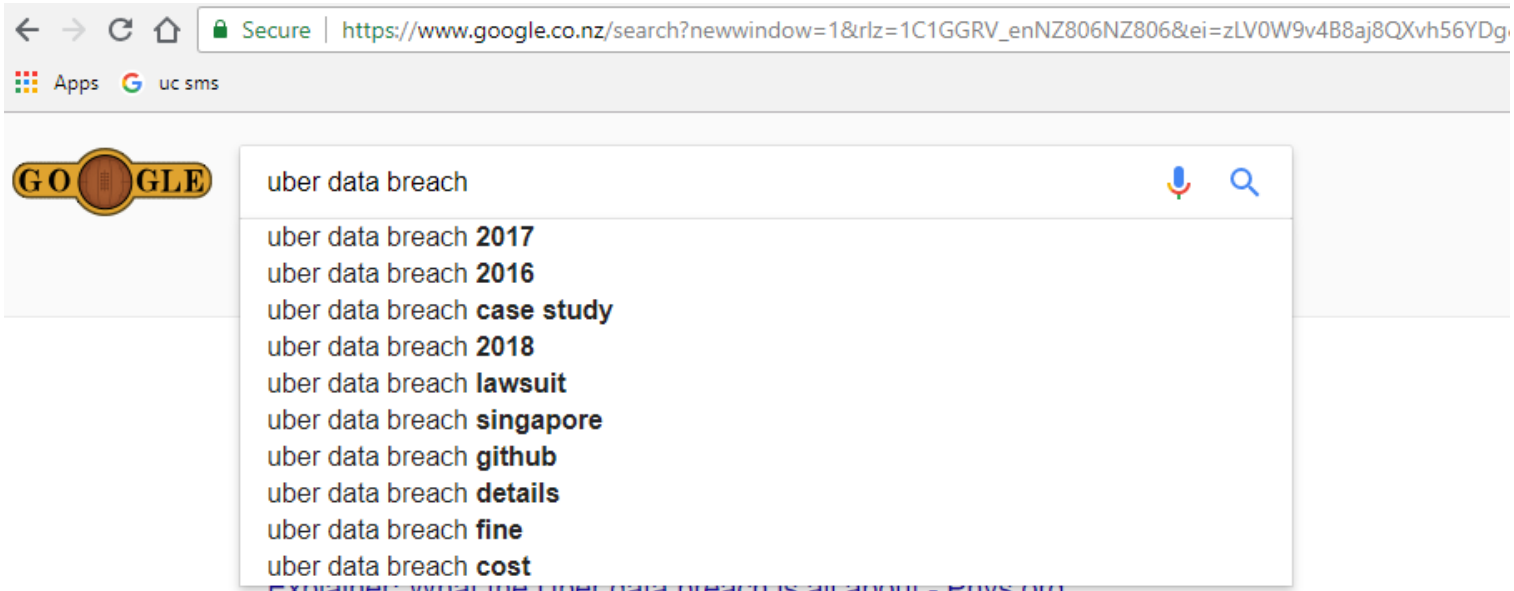
Uber hires chief security officer to help earn back user trust

by Sara Ashley O'Brien @saraashleyo

August 14, 2018: 2:00 PM ET

Recommend 85

Uber failed to disclose 2016 hack


Secure | https://www.google.co.nz/search?newwindow=1&rlz=1C1GGRV_enNZ806NZ806&ei=zLV0W9v4B8aj8QXvh56YDg
Apps
uc sms
GOOGLE
uber data breach
uber data breach 2017
uber data breach 2016
uber data breach case study
uber data breach 2018
uber data breach lawsuit
uber data breach singapore
uber data breach github
uber data breach details
uber data breach fine
uber data breach cost
Report inappropriate predictions
https://phys.org › Technology › Security
Nov 23, 2017 - Uber paid $100,000 to hackers who stole data on the ride-hailing company's ... such as trip details or credit card and Social Security numbers. ... While many security experts have criticized Uber for paying off the hackers with ...
Uber Paid Off Hackers to Hide Massive Data Breach - MIT Technology ...
https://www.technologyreview.com/.../uber-paid-off-hackers-to-hide-massive-data-bre...
Nov 22, 2017 - As with previous mega-hacks, more details will emerge in coming days and ... The breach raises big questions about the state of Uber's security ...
Uber concealed huge data breach - BBC News - BBC.com
https://www.bbc.com/news/technology-42075306
Nov 22, 2017 - The 2016 breach was hidden by the ride-sharing firm which paid hackers $100,000 (£75,000) to delete the data. ... The hackers found 57 million names, email addresses and mobile phone numbers, Uber said. Within that number, 600,000 drivers had their names and licence details exposed.
Uber says hackers behind 2016 data breach were in Canada, Florida ...
https://www.reuters.com/...uber.../uber-says-hackers-behind-2016-data-breach-were-i...
Feb 6, 2018 - The two people who hacked ride-hailing firm Uber's data in 2016 ... said Flynn, whose testimony described new details about the hack, the ...
Uber concealed massive hack that exposed data of 57m users and ...

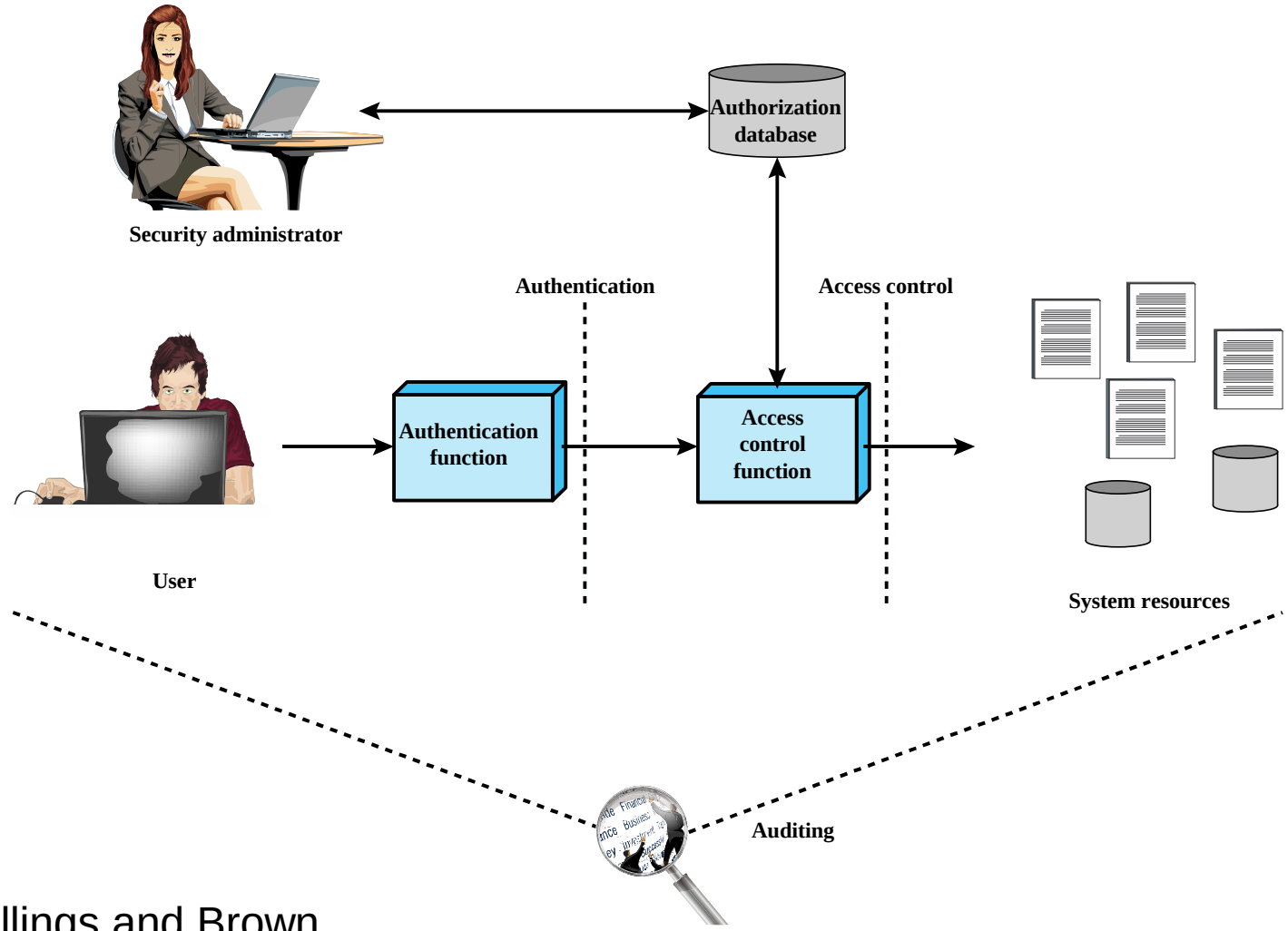
COSC 362

Chapter 4. Access Control

Access Control

- Subject, Object
- DAC (Discretionary Access Control)
- MAC (Mandatory Access Control)
  - Deployed primarily in military applications
  - Applies to OS, DBMS, etc.
  - Bell-LaPadula MLS (Multi-Level Security)
- Role-Based Access Control
  - IMO, not so widely used

Authentication vs Access Control



* Stallings and Brown

Figure 4.1 Relationship Among Access Control and Other Security Functions

Access Control Matrix

§ Was first introduced by Butler Lampson

MIT professor ‘made much of our world possible’

Computing luminaries celebrate the work of MIT adjunct professor and Turing Award winner Butler Lampson, one of the fathers of the modern PC.

Larry Hardesty, MIT News Office
February 18, 2014

Press Inquiries

Butler Lampson, an adjunct professor at MIT since 1987 and a technical fellow at Microsoft Research, has as good a claim as anyone to the title of “father of the modern PC.” As one of the founders of Xerox’s Palo Alto Research Center (PARC), Lampson helped create the Alto, the first computer to feature a mouse and a graphical user interface (GUI) — the progenitor of both the Apple Macintosh and the Windows operating system. He also led the development of Bravo, the first “what you see is what you get” — or WYSIWYG — word processor, which ran on the Alto.

RELATED

- Butler Lampson
- A list of computer systems that Lampson contributed to
- Lampson’s Turing Award citation

A.M. TURING AWARD WINNERS BY...

ALPHABETICAL LISTING | YEAR OF THE AWARD | RESEARCH SUBJECT

BIRTH:
December 23, 1943, Washington, D.C.

EDUCATION:
AB in Physics (Harvard, 1964); PhD in Electrical Engineering and Computer Science (University of California, Berkeley, 1967).

EXPERIENCE:
Xerox PARC (1971-1984); Digital Equipment Corporation, Corporate Consulting Engineer (1984-1995); Microsoft. Technical Fellow (from 1995)

BUTLER W LAMPSON

United States – 1992

CITATION

For contributions to the development of distributed, personal computing environments and the technology for their implementation: workstations, networks, operating systems, programming systems, displays, security and document publishing.

Access Control Matrix

SUBJECTS	OBJECTS: File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

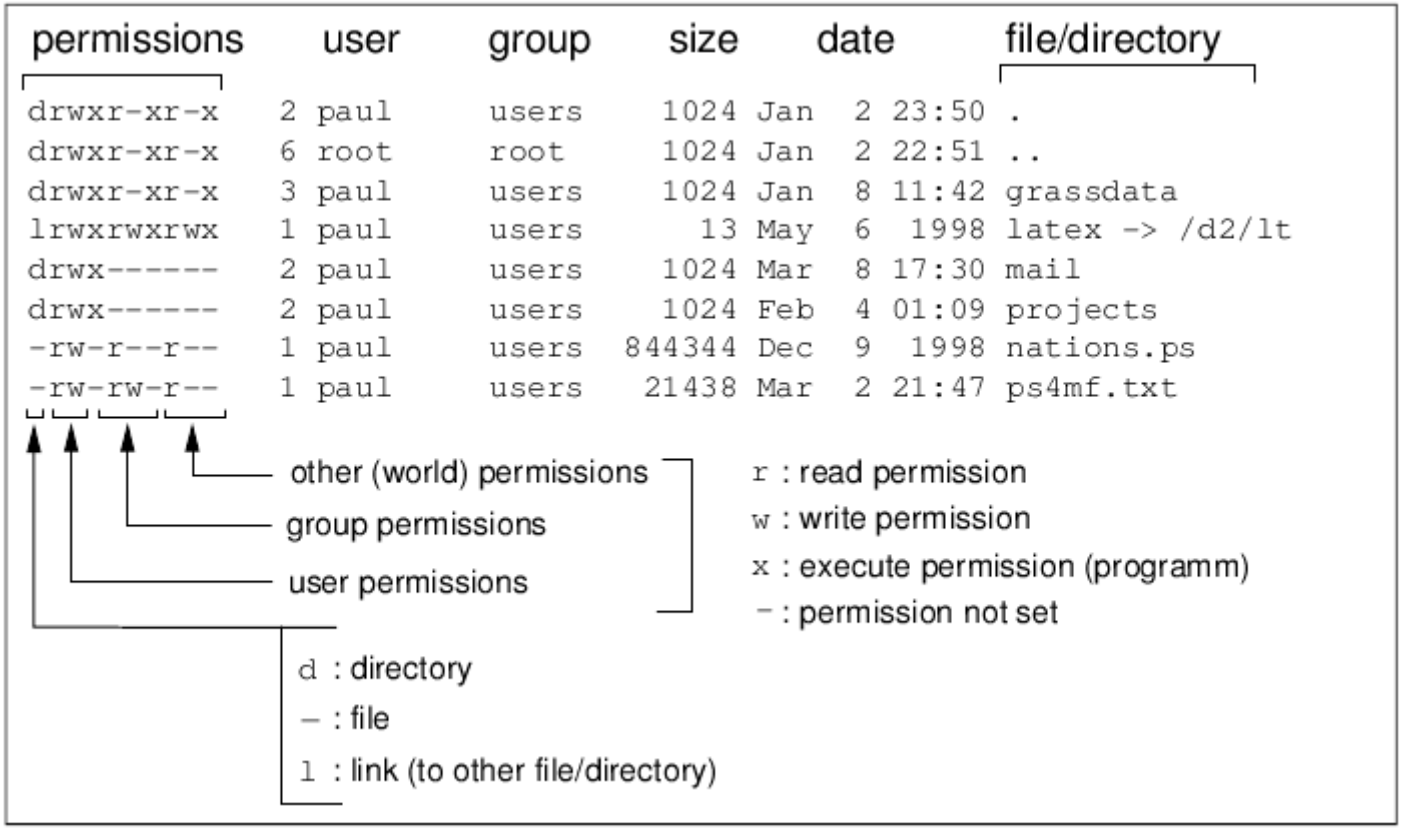
(a) Access matrix

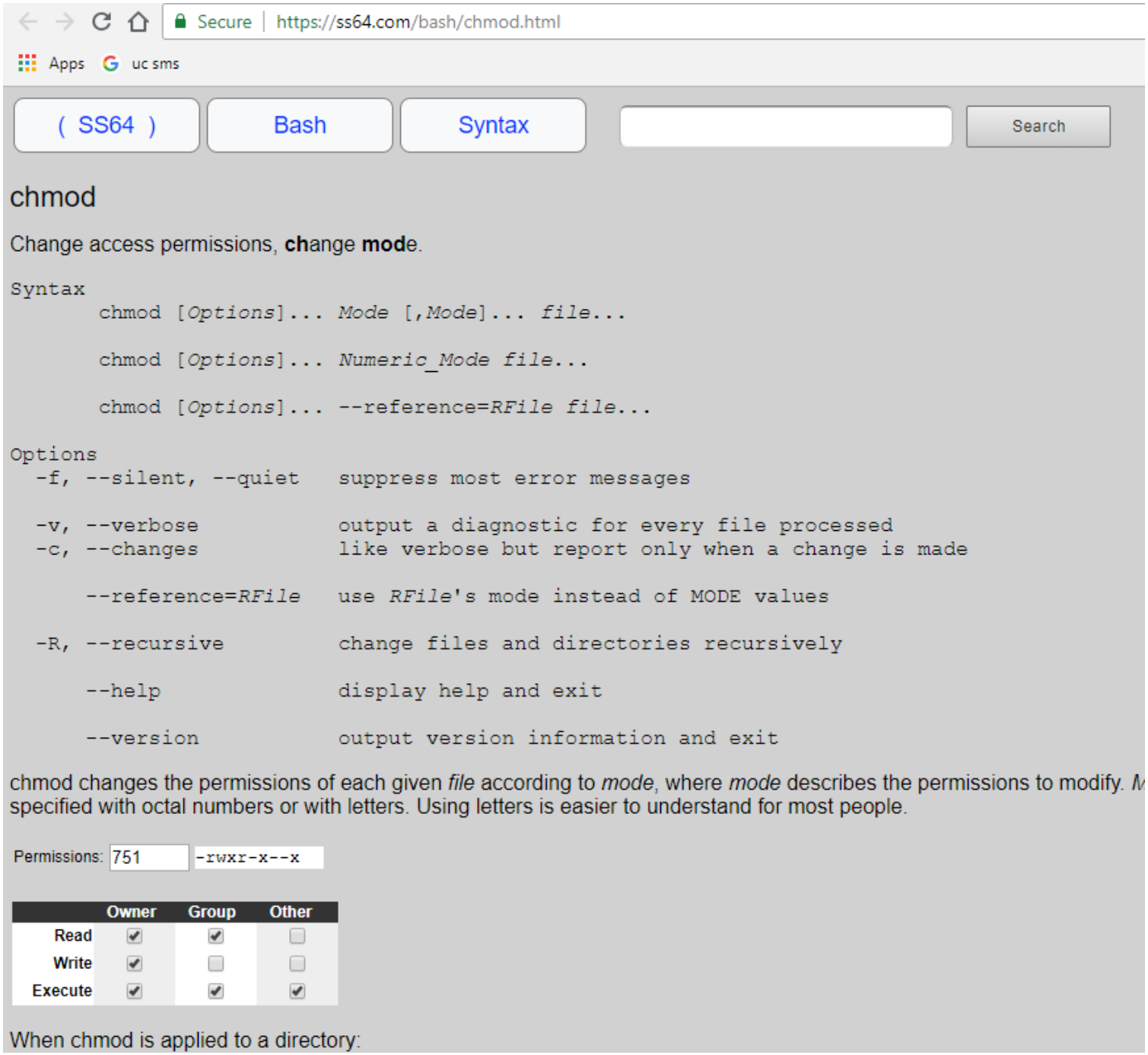
Subject, Object, and Access Rights

- Subject: An entity capable of accessing objects (e.g., process)
- Object: A resource to which access is controlled (e.g., file, directory, …)
  - Further divided into owner, group, world (other)
  - file's owner
  - users who are in the file's group
  - everybody else on the system (except the superuser/root)
- Access right (aka operation): Describes the way in which a subject may access an object (e.g., read, write, execute, …)

DAC in Unix

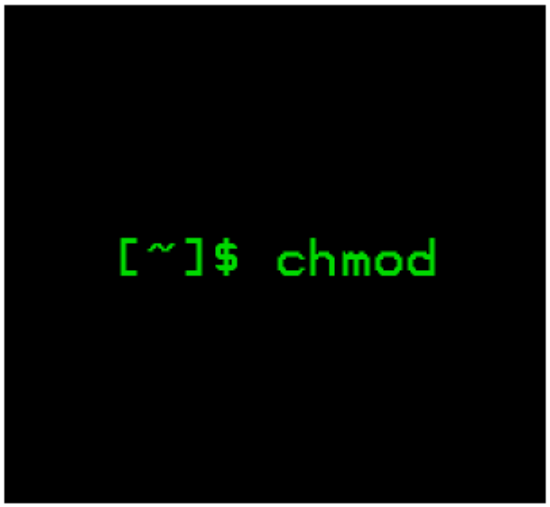
- Commonly known as file permission
  - chmod(). Numeric code is also available
  - UGO (user, group, other)




Secure | https://ss64.com/bash/chmod.html
Apps
uc sms
(SS64)
Bash
Syntax
Search
chmod
Change access permissions, change mode.
Syntax
chmod [Options]... Mode [,Mode]... file...
chmod [Options]... Numeric_Mode file...
chmod [Options]... --reference=RFile file...
Options
-f, --silent, --quiet suppress most error messages
-v, --verbose output a diagnostic for every file processed
-c, --changes like verbose but report only when a change is made
--reference=RFile use RFile's mode instead of MODE values
-R, --recursive change files and directories recursively
--help display help and exit
--version output version information and exit
chmod changes the permissions of each given file according to mode, where mode describes the permissions to modify. M
specified with octal numbers or with letters. Using letters is easier to understand for most people.
Permissions: 751 -rwxr-x--x
Owner Group Other
Read
Write
Execute
When chmod is applied to a directory:

Linux chmod command

Updated: 12/29/2017 by Computer Hope



ComputerHope.com

About chmod

chmod is used to change the permissions of files or directories.

Overview

On Linux and other Unix-like operating systems, there is a set of rules for each file which defines who can access that file, and how they can access it. These rules are called file permissions or file *modes*. The command name **chmod** stands for "change mode", and it is used to define the way a file can be accessed.

Before continuing, you should read the section What Are File Permissions, And How Do They Work? in our documentation of the **umask** command. It contains a comprehensive description of how to define and express file permissions.

Subject, Object, and Access Rights

- It is essential (and quite difficult) to accurately and completely identify all the subjects, objects, and access rights
  - Read
  - Write
  - Execute
  - Delete
  - Create
  - Search (e.g., change directory)
  - **Append?**
  - **Creating links (e.g., hard links, symbolic links)**
  - **Changing access rights or ownership?**

Unix File Permission

- Some file permission strings might be “valid or legal” but may make little logical sense
  - - --- --- --- (who can do what now?)
    - Can the superuser to chmod() on this?
  - - --- rwx ---
    - is owner also a group member?
  - - --- --- rwx
    - what is the precise definition of “other”?
- What does rwx mean when applied on directories?
  - Manual pages must be read carefully and thoroughly

PERMISSION	FILE	DIRECTORY
READ	CAN OPE AND READ CONTENT OF FILE	CAN LIST FILES PRESENT IN DIRECTORY AND CAN NOT READ FILES OF DIRECTORY
WRITE	USER CAN MODIFY CONTENTS OF FILE	USER CAN ADD OR DELETE FILES TO DIRECTORY
EXECUTE	USER CAN RUN EXECUTABLE FILES	USER CAN USE cd COMMAND AND CAN GO THROUGH THE DIRECTORY

*does write imply append?

https://wiki.archlinux.org/index.php/File_permissions_and_attributes

Viewing permissions

Use the **ls** command's `-l` option to view the permissions (or **file mode**) set for the contents of a directory, for example:

```
$ ls -l /path/to/directory
```

```
total 128
drwxr-xr-x 2 archie users  4096 Jul  5 21:03 Desktop
drwxr-xr-x 6 archie users  4096 Jul  5 17:37 Documents
drwxr-xr-x 2 archie users  4096 Jul  5 13:45 Downloads
-rw-rw-r-- 1 archie users  5120 Jun 27 08:28 customers.ods
-rw-r--r-- 1 archie users  3339 Jun 27 08:28 todo
-rwxr-xr-x 1 archie users  2048 Jul  6 12:56 myscript.sh
```

The first column is what we must focus on. Taking an example value of `drwxrwxrwx+`, the meaning of each character is explained in the following tables:

`d`	`rwx`	`rwx`	`rwx`	`+`
The file type, technically not part of its permissions. See `info ls -n "What information is listed"` for an explanation of the possible values.	The permissions that the owner has over the file, explained below.	The permissions that the group has over the file, explained below.	The permissions that all the other users have over the file, explained below.	A single character that specifies whether an alternate access method applies to the file. When this character is a space, there is no alternate access method. A `.` character indicates a file with a security context, but no other alternate access method. A file with any other combination of alternate access methods is marked with a `+` character, for example in the case of **Access Control Lists**.

Each of the three permission triads (`rwx` in the example above) can be made up of the following characters:

	Character	Effect on files	Effect on directories
Read permission (first character)	`-`	The file cannot be read.	The directory's contents cannot be shown.
	`r`	The file can be read.	The directory's contents can be shown.
Write permission (second character)	`-`	The file cannot be modified.	The directory's contents cannot be modified.
	`w`	The file can be modified.	The directory's contents can be modified (create new files or folders; rename or delete existing files or folders); requires the execute permission to be also set, otherwise this permission has no effect.
Execute permission (third character)	`-`	The file cannot be executed.	The directory cannot be accessed with **cd**.
	`x`	The file can be executed.	The directory can be accessed with **cd**; this is the only permission bit that in practice can be considered to be "inherited" from the ancestor directories, in fact if *any* folder in the path does not have the `x` bit set, the final file or folder cannot be accessed either, regardless of its permissions; see `path_resolution(7)` for more information.
	`s`	The **setuid** bit when found in the **u**ser triad; the **setgid** bit when found in the **g**roup triad; it is not found in the **o**thers triad; it also implies that `x` is set.	
	`S`	Same as `s`, but `x` is not set; rare on regular files, and useless on folders.	
	`t`	The **sticky** bit; it can only be found in the **o**thers triad; it also implies that `x` is set.	
	`T`	Same as `t`, but `x` is not set; rare on regular files, and useless on folders.	

`info Coreutils -n "Mode Structure"` and `chmod(1)` for more details.

Examples

Let us see some examples to clarify:

```
drwx------ 6 archie users  4096 Jul  5 17:37 Documents
```

Archie has full access to the Documents directory. He can list, create files and rename, delete any file in Documents, regardless of file permissions. His ability to access a file depends on the file's permission.

```
dr-x------ 6 archie users  4096 Jul  5 17:37 Documents
```

Archie has full access except he can not create, rename, delete any file. He can list the files and (if file's permission empowers) may access an existing file in Documents.

```
d-wx------ 6 archie users  4096 Jul  5 17:37 Documents
```

Archie can not do 'ls' in Documents but if he knows the name of an existing file then he may list, rename, delete or (if file's permission empowers him) access it. Also, he is able to create new files.

```
d--x------ 6 archie users  4096 Jul  5 17:37 Documents
```

Archie is only capable of (if file's permission empowers him) access those files in Documents which he knows of. He can not list already existing files or create, rename, delete any of them.

You should keep in mind that we elaborate on directory permissions and it has nothing to do with the individual file permissions. When you create a new file it is the directory that changes. That is why you need write permission to the directory.

Let us look at another example, this time of a file, not a directory:

```
-rw-r--r-- 1 archie users  5120 Jun 27 08:28 foobar
```

Here we can see the first letter is not `d` but `-`. So we know it is a file, not a directory. Next the owner's permissions are `rw-` so the owner has the ability to read and write but not execute. This may seem odd that the owner does not have all three permissions, but the `x` permission is not needed as it is a text/data file, to be read by a text editor such as Gedit, EMACS, or software like R, and not an executable in its own right (if it contained something like python programming code then it very well could be). The group's permssions are set to `r--`, so the group has the ability to read the file but not write/edit it in any way — it is essentially like setting something to read-only. We can see that the same permissions apply to everyone else as well.

superuser can do SO MUCH

- superuser (aka root) has an effective UID 0
  - process control
    - change priority of any process
    - set “hard limits” on resource usage (e.g., CPU file, file limit, etc)
  - device control
    - access any working device, shutdown or reboot, read or modify any memory location, create new device
  - network control
    - reconfigure network, run network services, etc
  - filesystem control
    - read, modify, or delete any file or program
    - add, create, or change user account
    - mount or unmount filesystem

Superuser

- But, even the superuser can't perform certain tasks such as
  - make a change to a filesystem mounted as read-only
  - unmount a filesystem that contains open files
  - decrypt the password stored in the shadow password file
- superuser is the main security weakness in UNIX operating system
  - some restrictions can be enforced
  - sulog is often provided
- Any user can become superuser with /bin/su program

Subject & Object: Not-so-apparent

- Processes, Files, Directories : no brainer
- Can processes be objects, too?
  - Need to understand exactly how interprocess communications occur
  - Exact semantics might be different from one implementation to another
- Hard link? Symbolic link?
- Device files?
- How about memory, buffer, etc?
  - becomes more complicated when both DAC and MAC are in place

https://www.ibm.com/developerworks/aix/library/au-spunix_sharedmemory/index.html

ks® Learn

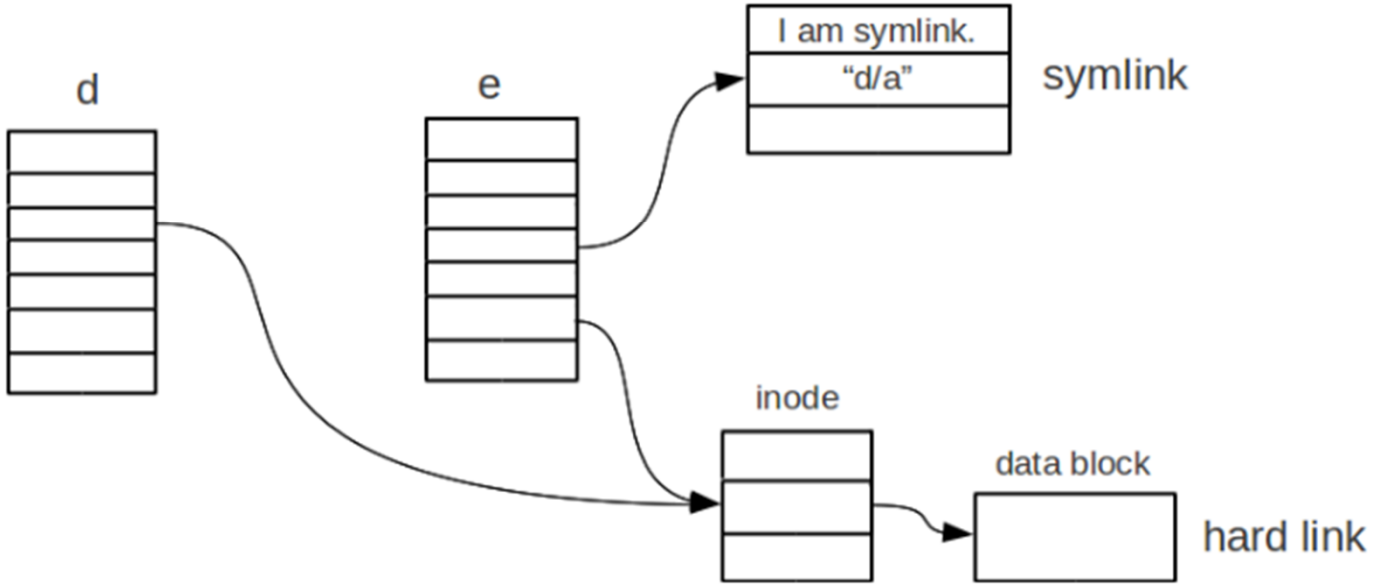
Table 1 below summarizes the forms of interprocess communication available on a typical UNIX system.

Table 1. Interprocess communication in UNIX

Name	Description	Scope	Use
File	Data is written to and read from a typical UNIX file. Any number of processes can interoperate.	Local	Sharing large data sets
Pipe	Data is transferred between two processes using dedicated file descriptors. Communication occurs only between a parent and child process.	Local	Simple data sharing, such as producer and consumer
Named pipe	Data is exchanged between processes via dedicated file descriptors. Communication can occur between any two peer processes on the same host.	Local	Producer and consumer, or command-and-control, as demonstrated with MySQL server and its command-line query utility
Signal	An interrupt alerts the application to a specific condition.	Local	Cannot transfer data in a signal, so mostly useful for process management
Shared memory	Information is shared by reading and writing from a common segment of memory.	Local	Cooperative work of any kind, especially if security is required.
Socket	After special setup, data is transferred using common input/output operations.	Local or remote	Network services such as FTP, ssh, and the Apache Web Server

Hard Link and Symbolic Link?

Before getting into the term Symbolic link and Hard link ,lets understand the term 'inode' . A Unix file is "stored" in two different parts of the disk—the data blocks and the inodes. The data blocks contain the "contents" of the file. But the **information about the file** is stored elsewhere—in the **inode**. Basically, inode is a file structure on a file system. More easily, it is a "database" of all file information except the file contents and the file name.Both the inodes and data blocks are stored in a "filesystem" which is how a disk partition is organized.



Symbolic links (aka soft links)?

How do I create soft link / symbolic link?

Soft links are created with the ln command. For example, the following would create a soft link named link1 to a file named file1, both in the current directory

```
$ ln -s file1 link1
```

To verify new soft link run:

```
$ ls -l file1 link1
```

Sample outputs:

```
-rw-r--r--  1 veryv  wheel  0 Mar  7 22:01 file1
lrwxr-xr-x  1 veryv  wheel  5 Mar  7 22:01 link1 -> file1
```

From the above outputs it is clear that a symbolic link named 'link1' contains the name of the file named 'file1' to which it is linked. So the syntax is as follows to create a symbolic link in Unix or Linux, at the shell prompt:

```
$ ln -s {source-filename} {symbolic-filename}
```

For example create a softlink for /webroot/home/httpd/test.com/index.php as /home/vivek/index.php, enter the following command:

```
$ ln -s /webroot/home/httpd/test.com/index.php /home/vivek/index.php
$ ls -l
```

Sample outputs:

```
lrwxrwxrwx 1 vivek  vivek    16 2007-09-25 22:53 index.php -> /webroot/h
```

- What if file permissons differ?
- What happens if I change the permission of either link or file itself?
- Must possess in-depth knowledge to perform security analysis

SUID and SGID

- It is a mechanism to allow unprivileged users to perform tasks (e.g., passwd, mail) that require privileges
  - Without ability to directly modify /etc/passwd file
- UNIX allows programs to be endowed with privileges. Processes executing these programs can assume another UID or GID when they are running
  - When a SUID program is run, its effective UID becomes that of the owner of the file, rather than the user who is running it.

Setuid and Security

setuid

From Wikipedia, the free encyclopedia

setuid and **setgid** (short for "**set u**ser **ID** upon execution" and "**set g**roup **ID** upon execution", respectively)[1] are Unix access rights flags that allow users to run an executable with the permissions of the executable's owner or group respectively and to change behaviour in directories. They are often used to allow users on a computer system to run programs with temporarily elevated privileges in order to perform a specific task. While the assumed user id or group id privileges provided are not always elevated, at a minimum they are specific.

`setuid` and `setgid` are needed for tasks that require higher privileges than those which common users have, such as changing their login password.[2] Some of the tasks that require elevated privileges may not immediately be obvious, though — such as the `ping` command, which must send and listen for control packets on a network interface.

Contents [hide]

Details Really Matter in Security

Excerpts from man pages about the sticky bit's effect on directories and files

Operating System	Directories	Files
AIX 5.2[1][*dead link*]	indicates that only file owners can link or unlink files in the specified directory.	sets the save-text attribute.
Solaris 11[2]	If a directory is writable and has S_ISVTX (the sticky bit) set, files within that directory can be removed or renamed only if one or more of the following is true (see unlink(2) and rename(2)): the user owns the file, the user owns the directory, the file is writable by the user, the user is a privileged user.	If a regular file is not executable and has S_ISVTX set, the file is assumed to be a swap file. In this case, the system's page cache will not be used to hold the file's data. If [...] set on any other file, the results are unspecified.
HP-UX[3]	If [...] set on a directory, an unprivileged user cannot delete or rename others' files in that directory.	[...] prevents the system from abandoning the swap-space image of the program-text portion of the file when its last user terminates. Then, when the next user of the file executes it, the text need not be read from the file system but can simply be swapped in, thus saving time.
Linux[4]	When [...] set on a directory, files in that directory may only be unlinked or renamed by root or the directory owner or the file owner.	the Linux kernel ignores the sticky bit on files.
FreeBSD[5]	If [...] set on a directory, an unprivileged user may not delete or rename files of other users in that directory.	The FreeBSD VM system totally ignores the sticky bit (ISVTX) for executables.
IRIX[6]	If [...] set on a directory, then any files created in that directory will take on the group ID of the directory rather than the group ID of the calling process. mount(1M) may be used to enable this feature regardless of the mode of the directory.	If the sticky bit, S_ISVTX, is set on a file that is a dynamic loader for an ELF executable, then when the executable is execed the old process's read only address spaces will be made available to the dynamic loader in the new process. This can improve program start up time considerably. The setting of the sticky bit on any other file has no effect.
Mac OS X (Leopard)[7]	A directory whose 'sticky bit' is set becomes an append-only directory [...] in which the deletion of files is restricted. A file in a sticky directory may only be removed or renamed by a user if the user has write permission for the directory and the user is the owner of the file, the owner of the directory, or the super-user. This feature is usefully applied to directories such as /tmp which must be publicly writable but should deny users the license to arbitrarily delete or rename each other's files. Any user may create a sticky directory.	The ISVTX (the sticky bit) has no effect on executable files. All optimization on whether text images remain resident in memory is handled by the kernel's virtual memory system.

chown, chgrp, ...

- File's owner and group can also be changed
- Use commands such as chmod, chown, chgrp with caution
- "Exact semantics" may differ
  - from one implementation to another
  - whether it is applied to a "regular" file, directory, symbolic link, etc

Message, IPC, Semaphore, Shared Memory, …

UnixWare's ACLs also allow specification of access rights to members of groups as defined to the system in the administrative file `/etc/group`. ACLs can be arbitrarily large; that is, the number of ACL entries is not limited by the system. The system administrator can set the maximum number of entries per ACL by setting a tunable parameter. (Naturally, as ACLs get larger, processing gets slower, which induces a practical limit on the number of ACL entries.)

Objects with ACLs

ACLs are associated with each filesystem object on a Secure File System (**sfs**) or Veritas File System (**vxfs**) and IPC object. ACLs for filesystem objects are stored in the associated inode. ACLs for IPC objects are stored in an internal structure associated with the instantiation of the IPC object.

DAC commands and system calls

The commands that a user can invoke to manipulate and read DAC permissions are

- **getacl**(1)
- **setacl**(1)

The system calls that a program can invoke to manipulate and read DAC permissions are

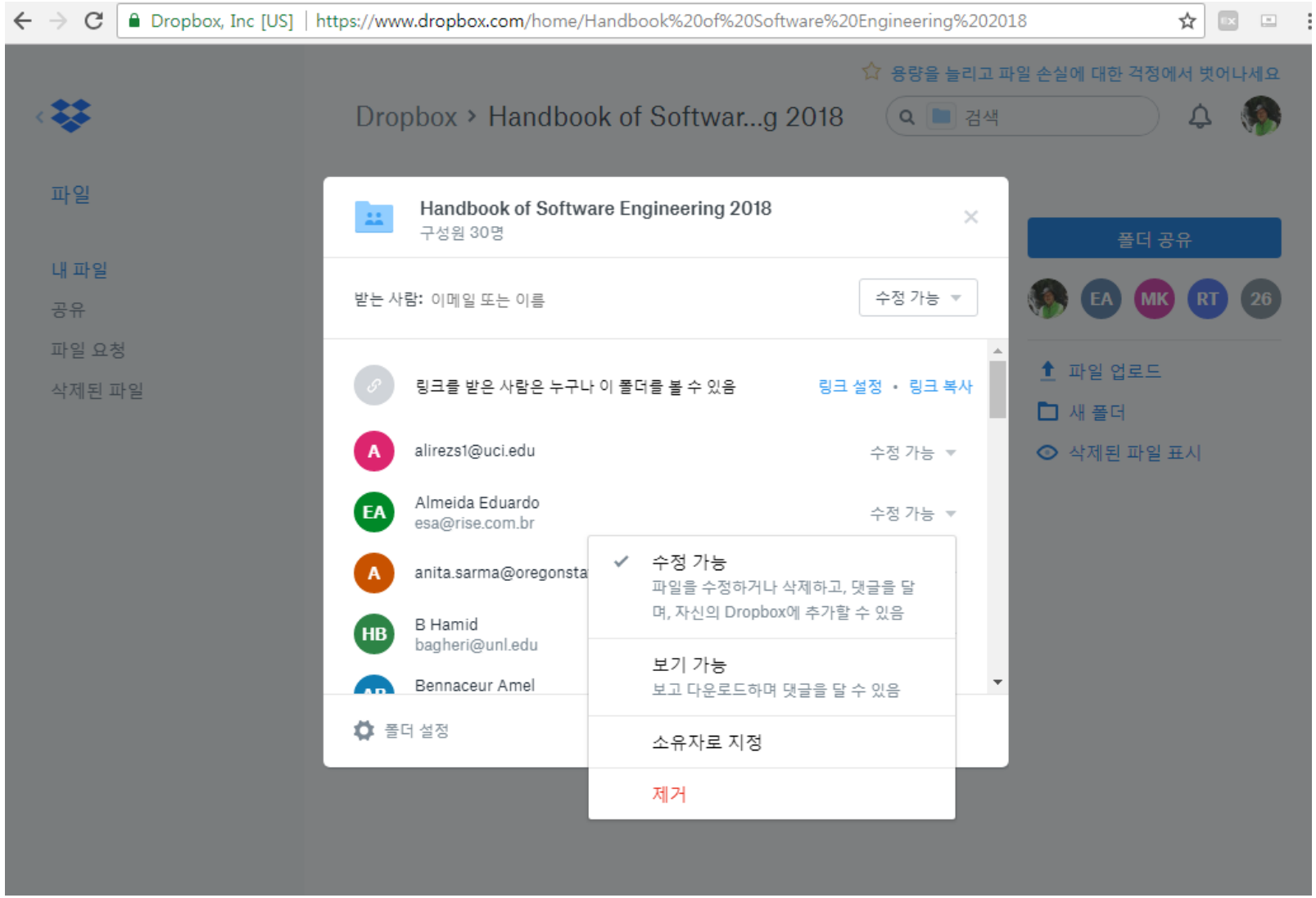
- **acl**(2)
- **aclipc**(2)
- **chmod**(2)
- **chown**(2)
- **msgctl**(2)
- **semctl**(2)
- **shmctl**(2)

The library function for reading and sorting ACL information is

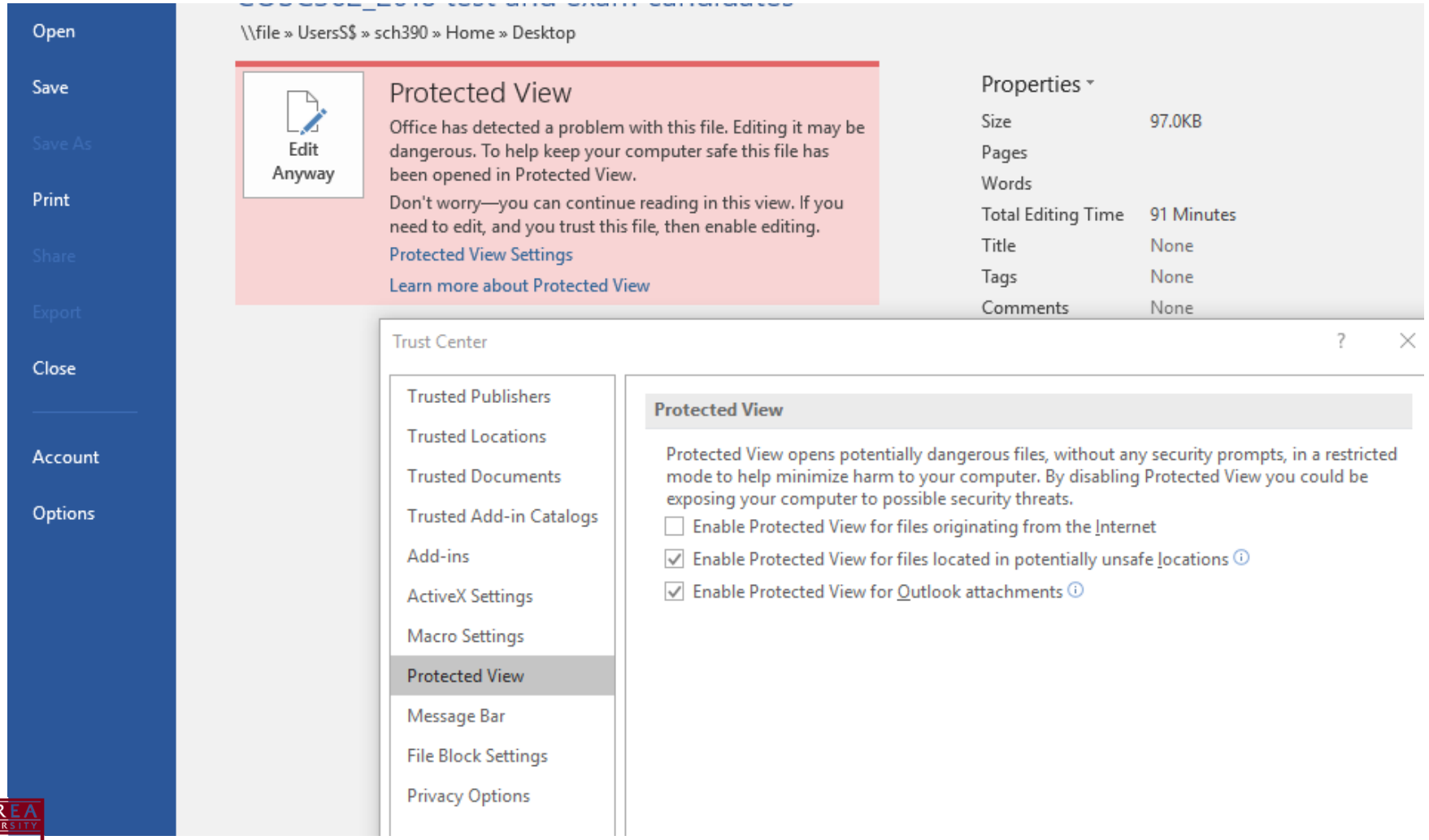
- **aclsort**(3C)

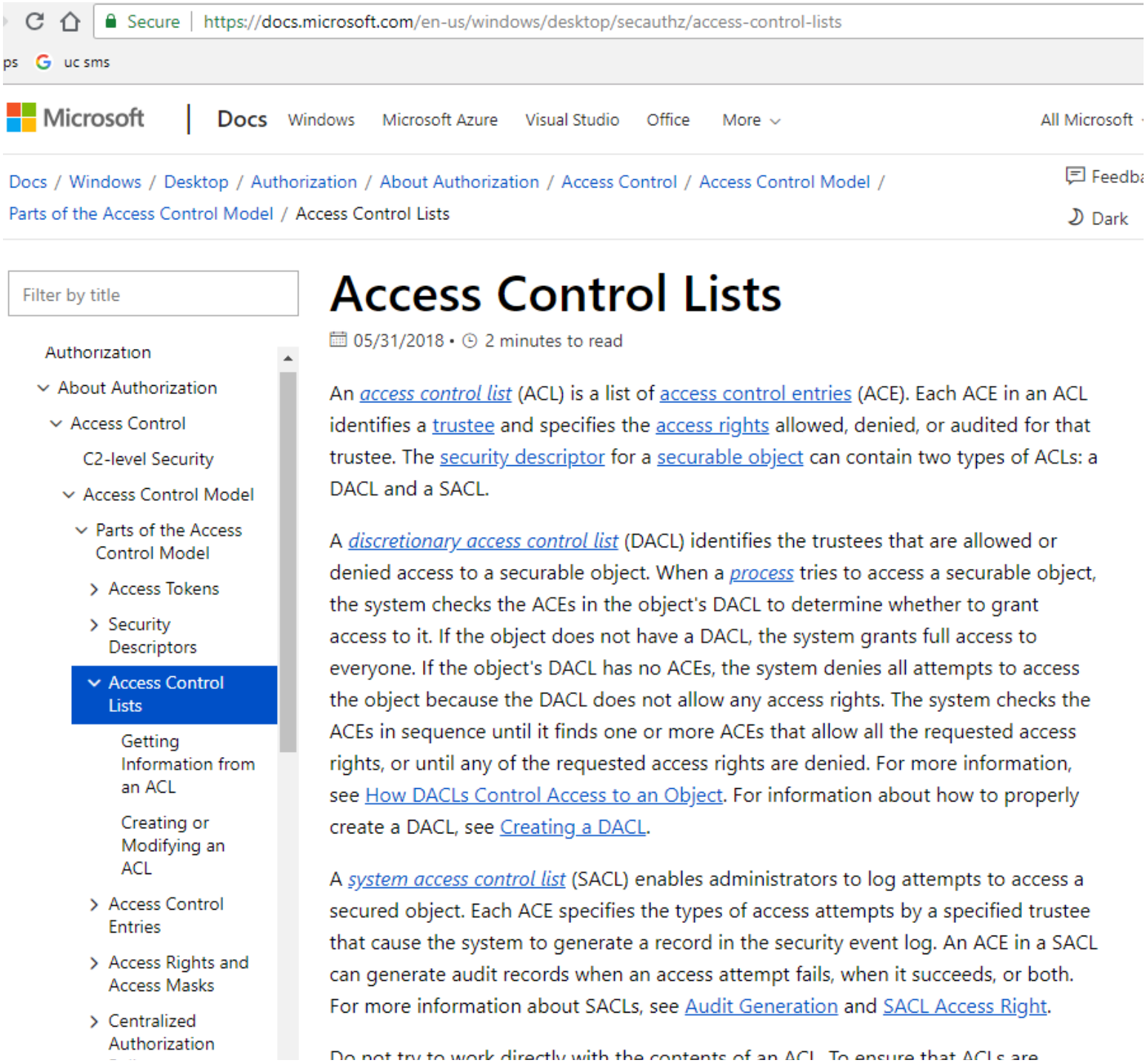
Access Control List is Widely Used

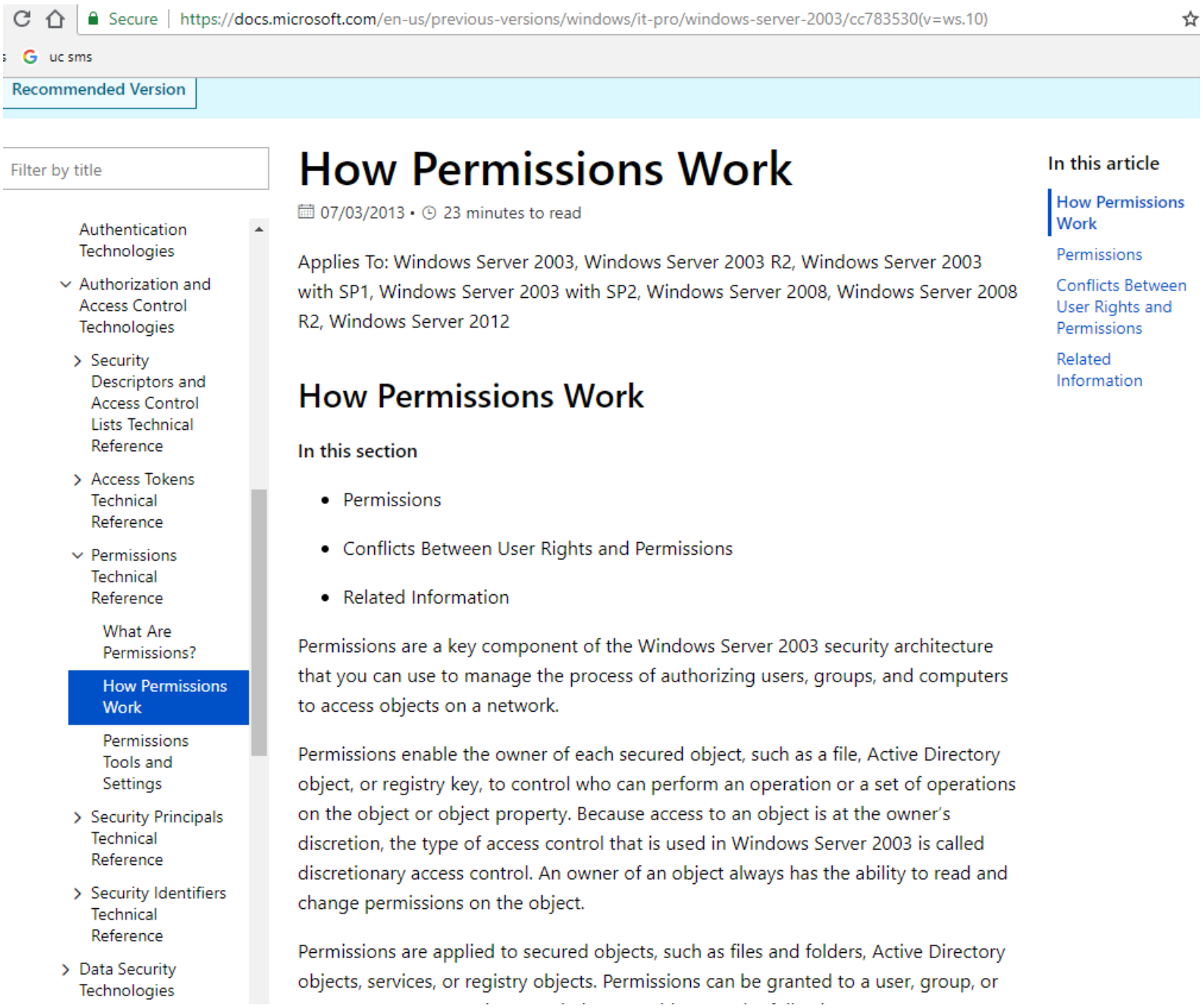
§ dropbox, google drive, pdf, ...



Another Example




Secure | https://docs.microsoft.com/en-us/windows/desktop/secauthz/access-control-lists
ps
uc sms
Microsoft | Docs Windows Microsoft Azure Visual Studio Office More
All Microsoft
Docs / Windows / Desktop / Authorization / About Authorization / Access Control / Access Control Model / Parts of the Access Control Model / Access Control Lists
Feedba
Dark
Filter by title
Authorization
About Authorization
Access Control
C2-level Security
Access Control Model
Parts of the Access Control Model
Access Tokens
Security Descriptors
Access Control Lists
Getting Information from an ACL
Creating or Modifying an ACL
Access Control Entries
Access Rights and Access Masks
Centralized Authorization
Access Control Lists
05/31/2018 • 2 minutes to read
An access control list (ACL) is a list of access control entries (ACE). Each ACE in an ACL identifies a trustee and specifies the access rights allowed, denied, or audited for that trustee. The security descriptor for a securable object can contain two types of ACLs: a DACL and a SACL.
A discretionary access control list (DACL) identifies the trustees that are allowed or denied access to a securable object. When a process tries to access a securable object, the system checks the ACEs in the object's DACL to determine whether to grant access to it. If the object does not have a DACL, the system grants full access to everyone. If the object's DACL has no ACEs, the system denies all attempts to access the object because the DACL does not allow any access rights. The system checks the ACEs in sequence until it finds one or more ACEs that allow all the requested access rights, or until any of the requested access rights are denied. For more information, see How DACLs Control Access to an Object. For information about how to properly create a DACL, see Creating a DACL.
A system access control list (SACL) enables administrators to log attempts to access a secured object. Each ACE specifies the types of access attempts by a specified trustee that cause the system to generate a record in the security event log. An ACE in a SACL can generate audit records when an access attempt fails, when it succeeds, or both. For more information about SACLs, see Audit Generation and SACL Access Right.


Secure | https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc783530(v=ws.10)
uc sms
Recommended Version
Filter by title
Authentication Technologies
Authorization and Access Control Technologies
Security Descriptors and Access Control Lists Technical Reference
Access Tokens Technical Reference
Permissions Technical Reference
What Are Permissions?
How Permissions Work
Permissions Tools and Settings
Security Principals Technical Reference
Security Identifiers Technical Reference
Data Security Technologies
How Permissions Work
07/03/2013 • 23 minutes to read
Applies To: Windows Server 2003, Windows Server 2003 R2, Windows Server 2003 with SP1, Windows Server 2003 with SP2, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012
How Permissions Work
In this section
Permissions
Conflicts Between User Rights and Permissions
Related Information
Permissions are a key component of the Windows Server 2003 security architecture that you can use to manage the process of authorizing users, groups, and computers to access objects on a network.
Permissions enable the owner of each secured object, such as a file, Active Directory object, or registry key, to control who can perform an operation or a set of operations on the object or object property. Because access to an object is at the owner's discretion, the type of access control that is used in Windows Server 2003 is called discretionary access control. An owner of an object always has the ability to read and change permissions on the object.
Permissions are applied to secured objects, such as files and folders, Active Directory objects, services, or registry objects. Permissions can be granted to a user, group, or
In this article
How Permissions Work
Permissions
Conflicts Between User Rights and Permissions
Related Information

ACL Default Setting

§ You must carefully review trade-offs and decide

ORACLE

Solaris Common Desktop Environment: User's Guide

« **Previous**: To Change an Access Control List Entry **Next**: To Set Required Default Entry Types »

Setting Default Permissions Through an Access Control List

When you create a file or folder within a folder, it inherits the basic permissions set by the system administrator. (To determine the current defaults, create a new file or folder and then choose Properties from the Selected menu to view the permissions.)

You can use an Access Control List to set default basic permissions yourself for any file or folder that is created within a folder. The ACL for that folder must contain entries for all four of the following **required** Default entry types: Default Owning User, Default Owning Group, Default Other, and Default Mask. An ACL can contain only one entry of each required type.

The file or folder inherits the values for Owner, Group, and Other from the person who creates it and inherits the basic permissions from the required ACL Default entry types on the containing folder. ACL entries of these types do not have names associated with them.

You can also set **optional** Default entry types—Default User and Default Group— for any file or folder that is created within a folder. You can create as many Default User or Default Group ACL entries as you want. You must specify the name of the user or group when you create the ACL entry.

Any ACL in which you want to put a Default User or Default Group entry must also contain one of each required entry type.

ACL Default Setting

- Systems are likely to choose similar setting, but you must verify to be sure

Example

Suppose that the values for Owner and Group for a user named Carla are `otto` and `otto_staff`, respectively. The value for Other (call it `otto_other`) is everyone at Carla's company except for Carla and the members of `otto_staff`. Carla creates these required Default ACLs on her folder named `Project1`:

- Default Owning User with permissions rwx (read, write, execute)
- Default Owning Group with permissions rx (read, execute)
- Default Other with permissions no-read, no-write, no-execute
- Default Mask with permissions rw (read, write)

Any file or folder subsequently placed in the `Project1` folder inherits these basic permissions from `Project1`:

- The file or folder Owner value is `otto` and `otto` has read, write, and execute permission on that file or folder
- The file or folder Group value is `otto_staff` and `otto_staff` has read and execute permission on that file or folder
- The file or folder Other value is `otto_other` and `otto_other` has no-read, no-write, and no-execute permission on that file or folder

Also, the file or folder has a Mask entry in the Access Control List Permissions scrolling list with the value rw (read, write).

If Carla also adds an optional ACL of type Default User (Default Group) for the `Project1` folder, then any file or folder subsequently placed in `Project1` will inherit an ACL of type User (Group).

« **Previous**: To Change an Access Control List Entry **Next**: To Set Required Default Entry Types »

Mandatory Access Control (MAC)

https://en.wikipedia.org/wiki/Mandatory_access_control

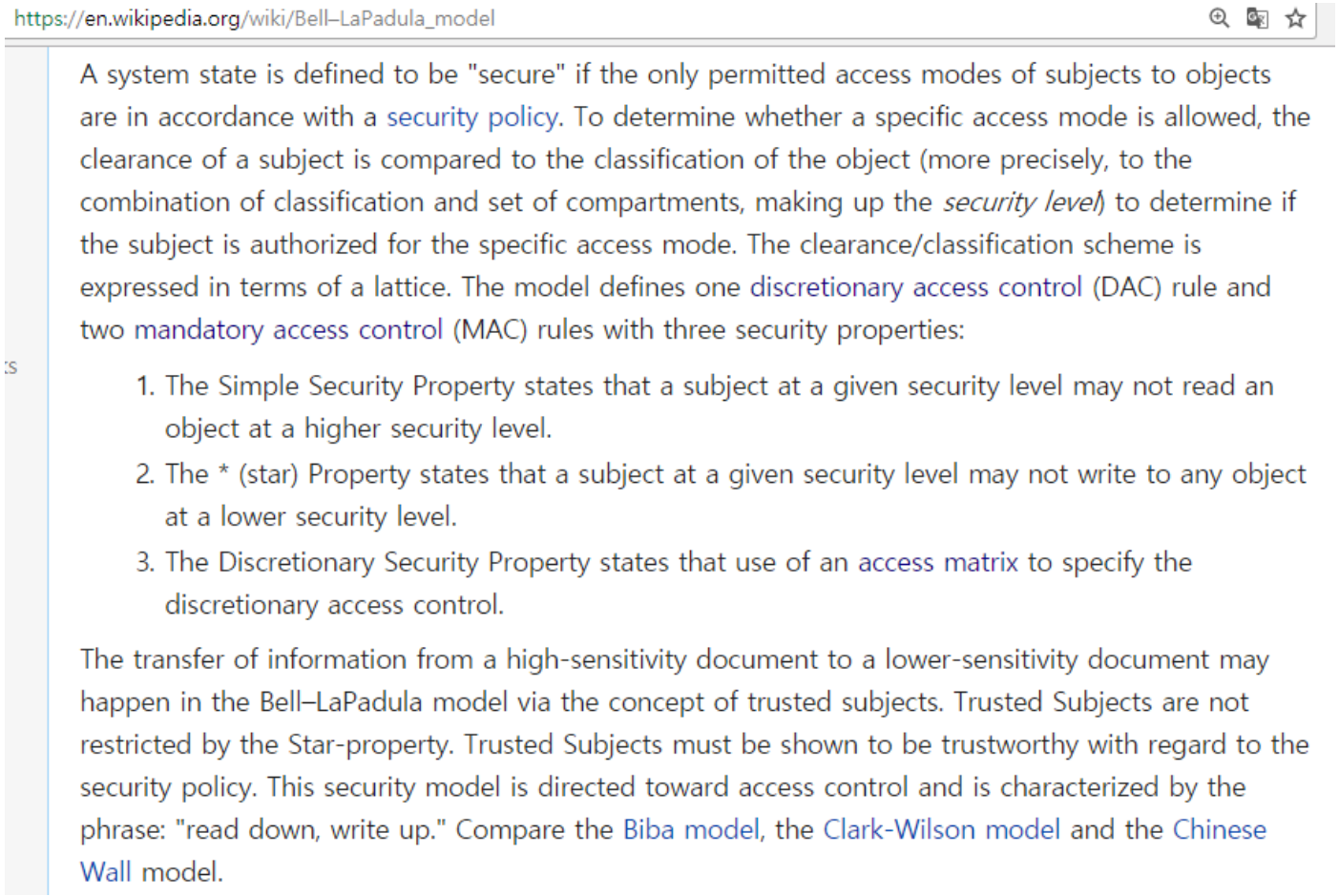
access control; in this case, the objects are tables, views, procedures, etc.

With mandatory access control, this security policy is centrally controlled by a security policy administrator; users do not have the ability to override the policy and, for example, grant access to files that would otherwise be restricted. By contrast, discretionary access control (DAC), which also governs the ability of subjects to access objects, allows users the ability to make policy decisions and/or assign security attributes. (The traditional Unix system of users, groups, and read-write-execute permissions is an example of DAC.) MAC-enabled systems allow policy administrators to implement organization-wide security policies. Under MAC (and unlike DAC), users cannot override or modify this policy, either accidentally or intentionally. This allows security administrators to define a central policy that is guaranteed (in principle) to be enforced for all users.

Historically and traditionally, MAC has been closely associated with multi-level security (MLS) and specialized military systems. In this context, MAC implies a high degree of rigor to satisfy the constraints of MLS systems. More recently, however, MAC has evolved out of the MLS niche and has started to become more mainstream. The more recent MAC implementations, such as SELinux and AppArmor for Linux and Mandatory Integrity Control for Windows, allow administrators to focus on issues such as network attacks and malware without the rigor or constraints of MLS.

MLS and Bell-LaPadula

Multilevel security or **multiple levels of security** (**MLS**) is the application of a computer system to process information with incompatible classifications (i.e., at different security levels), permit access by users with different security clearances and needs-to-know, and prevent users from obtaining access to information for which they lack authorization. There are two contexts for the use of Multilevel


https://en.wikipedia.org/wiki/Bell–LaPadula_model

A system state is defined to be "secure" if the only permitted access modes of subjects to objects are in accordance with a security policy. To determine whether a specific access mode is allowed, the clearance of a subject is compared to the classification of the object (more precisely, to the combination of classification and set of compartments, making up the *security level*) to determine if the subject is authorized for the specific access mode. The clearance/classification scheme is expressed in terms of a lattice. The model defines one discretionary access control (DAC) rule and two mandatory access control (MAC) rules with three security properties:

1. The Simple Security Property states that a subject at a given security level may not read an object at a higher security level.
2. The * (star) Property states that a subject at a given security level may not write to any object at a lower security level.
3. The Discretionary Security Property states that use of an access matrix to specify the discretionary access control.

The transfer of information from a high-sensitivity document to a lower-sensitivity document may happen in the Bell–LaPadula model via the concept of trusted subjects. Trusted Subjects are not restricted by the Star-property. Trusted Subjects must be shown to be trustworthy with regard to the security policy. This security model is directed toward access control and is characterized by the phrase: "read down, write up." Compare the Biba model, the Clark-Wilson model and the Chinese Wall model.

MLS Policy Definition

- Generally, "No read up, No write down."
- Subtle variations may exist from one implementation to another
  - Is it a security violation to allow "write up"?
  - Why are covert channels security threats?

With Bell-LaPadula, users can create content only at or above their own security level (i.e. secret researchers can create secret or top-secret files but may not create public files; no write-down). Conversely, users can view content only at or below their own security level (i.e. secret researchers can view public or secret files, but may not view top-secret files; no read-up).

The Bell–LaPadula model explicitly defined its scope. It did not treat the following extensively:

- Covert channels. Passing information via pre-arranged actions was described briefly.
- Networks of systems. Later modeling work did address this topic.
- Policies outside multilevel security. Work in the early 1990s showed that MLS is one version of boolean policies, as are all other published policies.

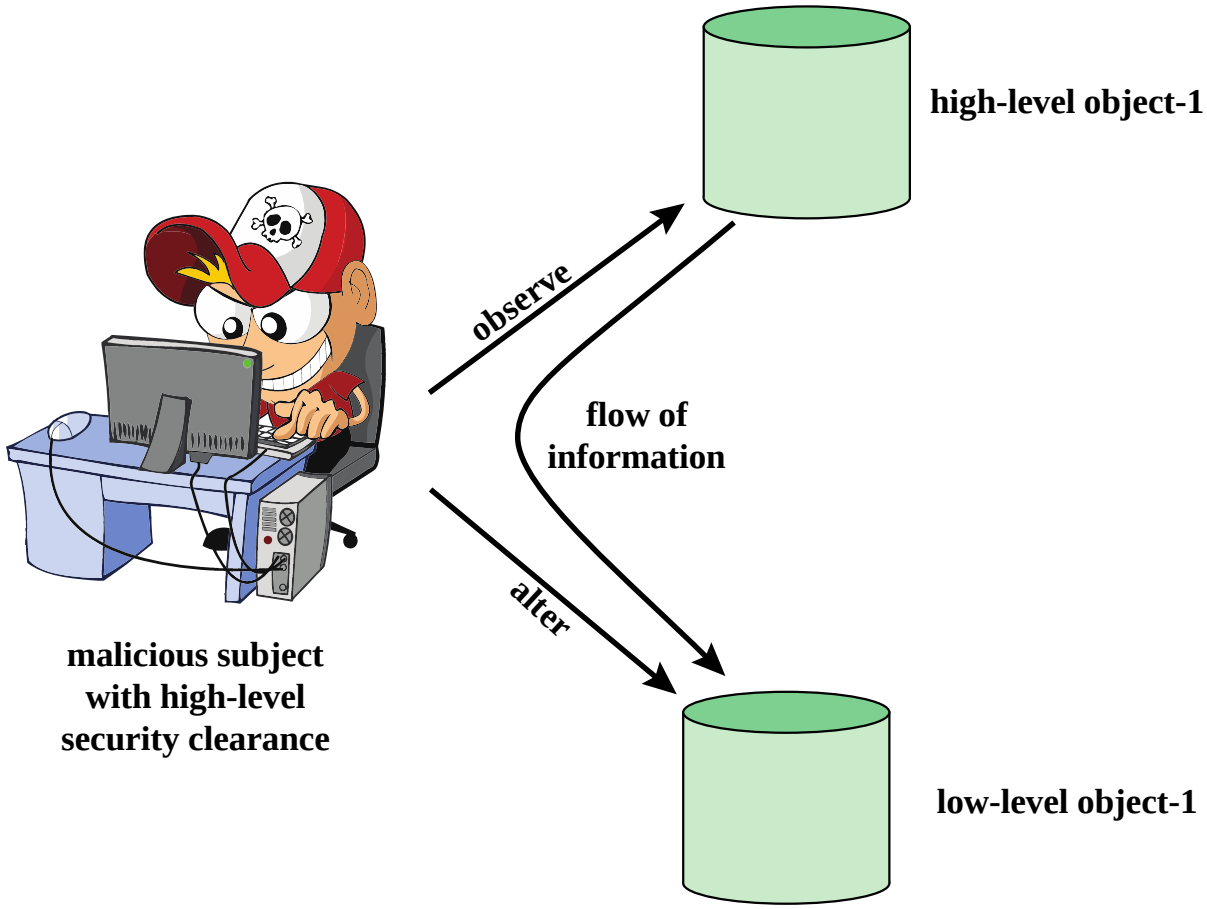


Figure 13.1 Information Flow Showing the Need for the *-property

MLS and Security Levels

https://www.centos.org/docs/5/html/Deployment_Guide-en-US/sec-mls-ov.html

rights.

43.6.2. Security Levels, Objects and Subjects

As discussed above, subjects and objects are labeled with *Security Levels* (SLs), which are composed of two types of entities:

1. `Sensitivity:` — A hierarchical attribute such as "Secret" or "Top Secret".

2. `Categories:` — A set of non-hierarchical attributes such as "US Only" or "UFO".

An SL must have one sensitivity, and may have zero or more categories.

Examples of SLs are: { Secret / UFO, Crypto }, { Top Secret / UFO, Crypto, Stargate } and { Unclassified }

Note the hierarchical sensitivity followed by zero or more categories. The reason for having categories as well as sensitivities is so that sensitivities can be further compartmentalized on a need-to-know basis. For example, while a process may be cleared to the "Secret" sensitivity level, it may not need any type of access to the project "Warp Drive" (which could be the name of a category).

MLS Policy

43.6.3. MLS Policy

SELinux uses the *Bell-La Padula* BLP model, with Type Enforcement (TE) for integrity. In simple terms, MLS policy ensures that a Subject has an appropriate clearance to access an Object of a particular classification.

For example, under MLS, the system needs to know how to process a request such as: Can a process running with a clearance of { Top Secret / UFO, Rail gun } write to a file classified as { Top Secret / UFO } ?

The MLS model and the policy implemented for it will determine the answer. (Consider, for example, the problem of information leaking out of the Rail gun category into the file).

MLS meets a very narrow (yet critical) set of security requirements based around the way information and personnel are managed in rigidly controlled environments such as the military. MLS is typically difficult to work with and does not map well to general-case scenarios.

Type Enforcement (TE) under SELinux is a more flexible and expressive security scheme, which is in many cases more suitable than MLS.

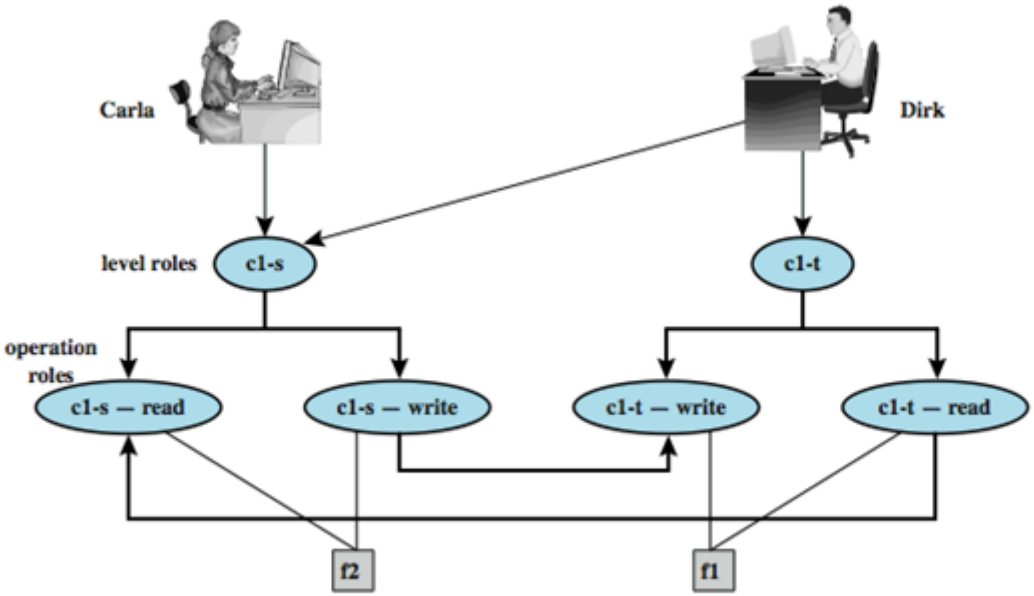
There are, however, several scenarios where traditional MLS is still required. For example, a file server where the stored data may be of mixed classification and where clients connect at different clearances. This results in a large number of Security Levels and a need for strong isolation all on a single system.

This type of scenario is the reason that SELinux includes MLS as a security model, as an adjunct to TE.

DAC and MAC

- Study Chapter 13, section 1 of the textbook
  - ss-property (“no read up”)
  - *-property (“no write down”)
  - ds-property
    - An individual may grant to another individual access to a document based on the owner’s discretion, constrained by the MAC rules
    - In other words, both DAC and MAC tests must pass to access an object.

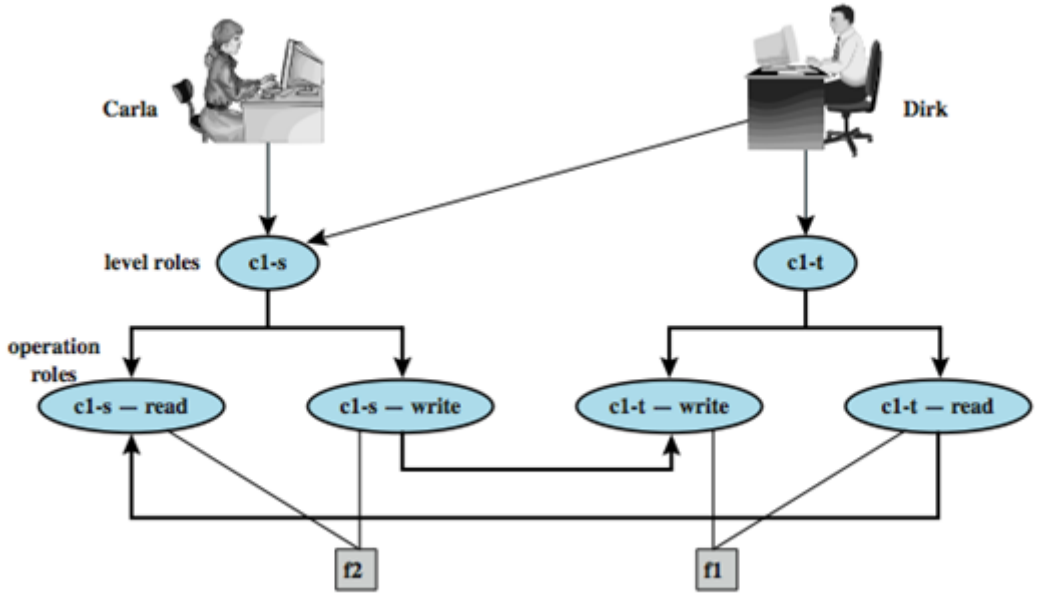
BLP Example



(a) Two new files are created: f1: c1-t; f2: c1-s

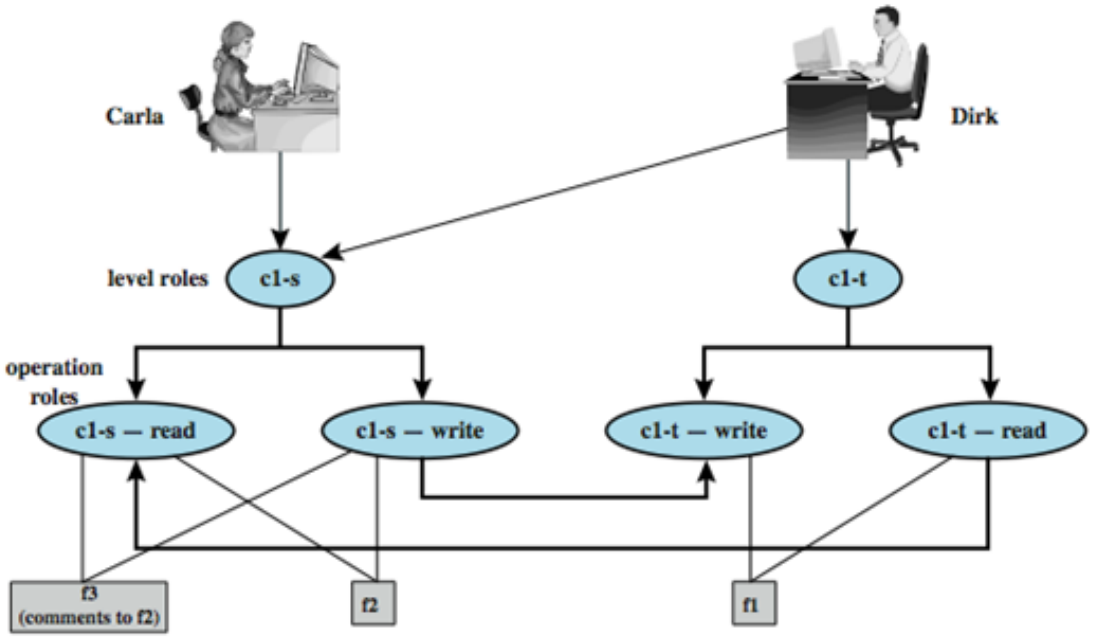
An example illustrates the operation of the BLP model, and also highlights a practical issue that must be addressed. We assume a role-based access control system. Carla and Dirk are users of the system. Carla is a student (s) in course c1. Dirk is a teacher (t) in course c1, but may also access the system as a student; thus two roles are assigned to Dirk: Carla: (c1-s); and Dirk: (c1-t), (c1-s).
The student role is assigned a lower security clearance and the teacher role a higher security clearance. Let us look at some possible actions:

Source: Stallings and Brown, section 13.1



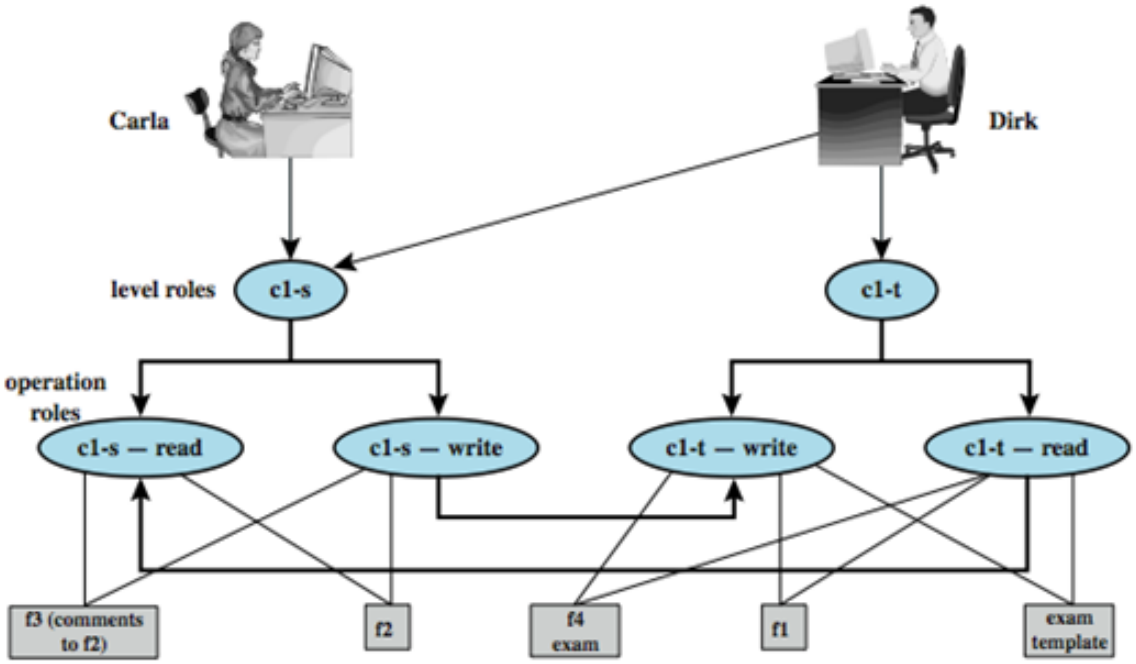
(a) Two new files are created: f1: c1-t; f2: c1-s

1. Dirk creates a new file f1 as c1-t; Carla creates file f2 as c1-s (Figure 10.2a). Carla can read and write to f2, but cannot read f1, because it is at a higher classification level (teacher level). In the c1-t role, Dirk can read and write f1 and can read f2 if Carla grants access to f2. However, in this role, Dirk cannot write f2 because of the *-property; neither Dirk nor a Trojan horse on his behalf can downgrade data from the teacher level to the student level. Only if Dirk logs in as a student can he create a c1-s file or write to an existing c1-s file, such as f2. In the student role, Dirk can also read f2.



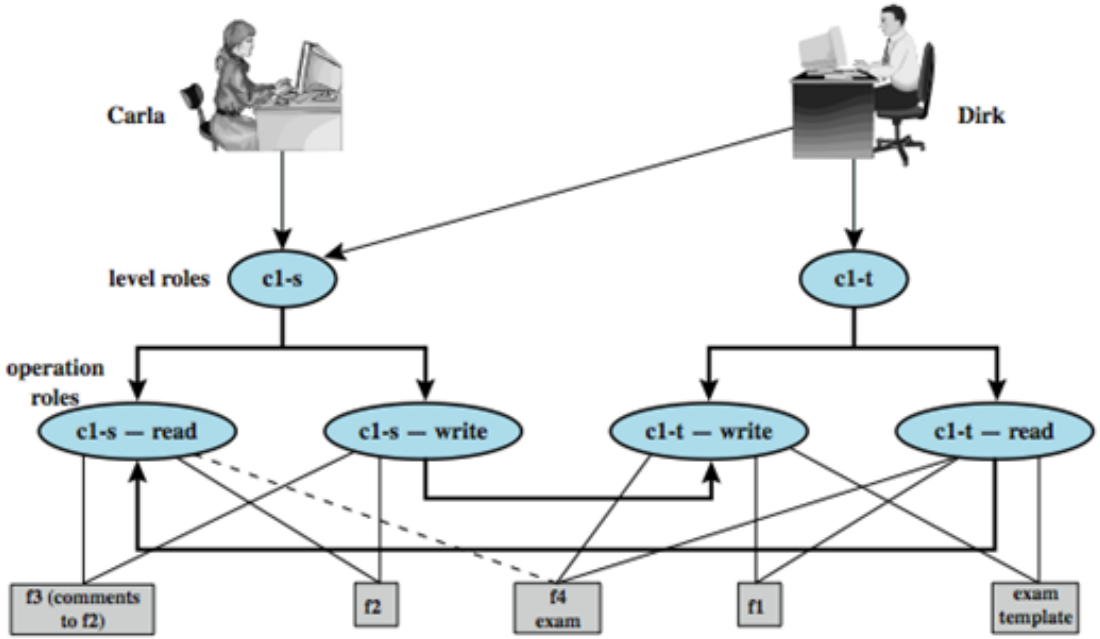
(b) A third file is added: f3: c1-s

2. Dirk reads f2 and wants to create a new file with comments to Carla as feedback. Dirk must sign in student role c1-s to create f3 so that it can be accessed by Carla (Figure 10.2b). In a teacher role, Dirk cannot create a file at a student classification level.



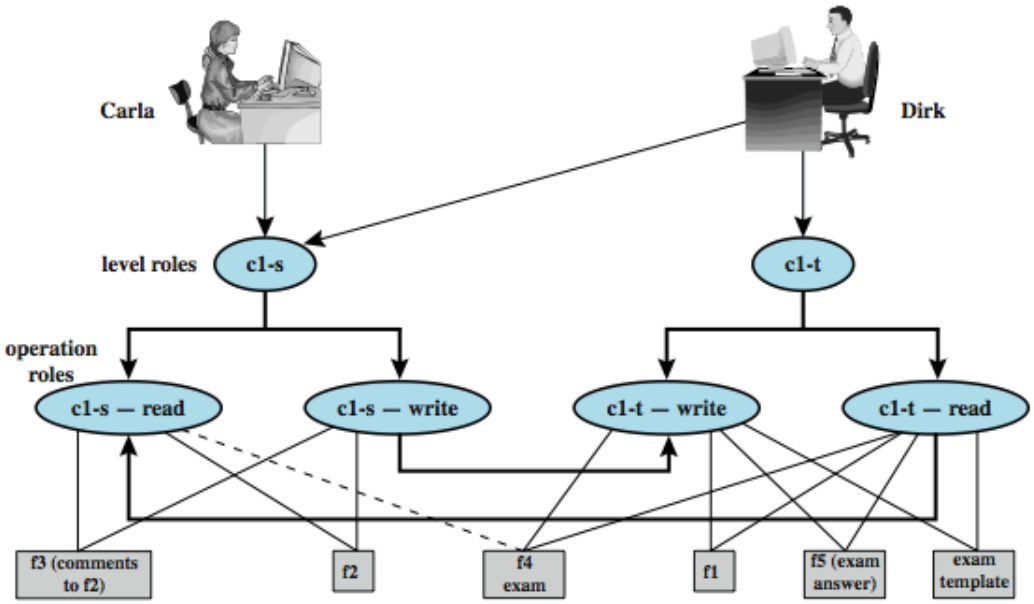
(c) An exam is created based on an existing template: f4: c1-t

3. Dirk creates an exam based on an existing template file store at level c1-t. Dirk must log in as c1-t to read the template and the file he creates (f4) must also be at the teacher level (Figure 10.2c).



(d) Carla, as student, is permitted acess to the exam: f4: c1-s

4. Dirk wants Carla to take the exam and so must provide her with read access. However, such access would violate the ss-property. Dirk must downgrade the classification of f4 from c1-t to c1-s. Dirk cannot do this in the c1-t role because this would violate the *-property. Therefore, a security administrator (possibly Dirk in this role) must have downgrade authority and must be able to perform the downgrade outside the BLP model. The dotted line in Figure 10.2d connecting f4 with c1-s-read indicates that this connection has not been generated by the default BLP rules but by a system operation.



(e) The answers given by Carla are only accessible for the teacher: f5: c1-t

5. Carla writes the answers to the exam into a file f5. She creates the file at level c1-t so that only Dirk can read the file. This is an example of writing up, which is not forbidden by the BLP rules. Carla can still see her answers at her workstation but cannot access f5 for reading.

This discussion illustrates some critical practical limitations of the BLP model. First, as noted in step 4, the BLP model has no provision to manage the “downgrade” of objects, even though the requirements for multilevel security recognize that such a flow of information may be required, provided it reflects the will of an authorized user. Hence, any practical implementation of a multilevel system has to support such a process in a controlled and monitored manner. Related to this is another concern. A subject constrained by the BLP model can only be “editing” (reading and writing) a file at one security level while also viewing files at the same or lower levels. If the new document consolidates information from a range of sources and levels, some of that information is now classified at a higher level than it was originally. This is known as classification creep, and is a well-known concern with multilevel information.

Role-based Access Control

- RBAC is based on the roles that users assume in a system rather than the user's identity
  - Roles can be assigned statically or dynamically
- DAC and RBAC are NOT, IMHO, fundamentally different (e.g., "root" or "super user")
  - RBAC is not widely used in practice

Access Control is more than permission bits

Apple, Samsung, and LG back plans to replace car keys with smartphones

Chris Mills @chrisfmills
June 21st, 2018 at 4:31 PM

Share Tweet

Thanks to contactless payments and mobile wallets, smartphones are already making a bid to replace physical credit cards — and, one day, your driver's license. Next up on the list of everyday items to be digitized might be your car key, thanks to a new standard backed by smartphone giants like Apple and Samsung, and car companies including Audi, VW, GM, and Hyundai.

The standard is called Digital Key Release 1.0, and it's designed to use the NFC chip inside modern smartphones to unlock and turn on modern vehicles. The first version of the specification will require dealers to transfer ownership of the digital key to smartphones, after which owners will be able to do everything they can currently do with a physical key.

The Digital Key interface allows "drivers to lock, unlock, start the engine, and share access to their car – all from their smart devices" said the Connected Car Consortium, the group behind the specification. It uses NFC to communicate between the car and the smart device, and relies on the "secure element" — the same physical chip that authenticates mobile wallet transactions — to keep things safe. Version 1.0 is available to member companies now, and the group is already working on a version 2.0, which will allow companies to integrate it more simply and with fewer development costs.

Google digital key connected car

All Images News Videos More Settings Tools

About 129,000,000 results (0.44 seconds)

Securely Share Vehicles with our Digital Car Key - Ericsson

https://www.ericsson.com/en/internet.../automotive/connected-vehicle.../digital-car-ke...

Our **Digital Car Key** allows people to securely share **vehicles** via a smartphone ... of **cars** as **connected** devices is drastically changing the ways we use **vehicles**.

People also search for

virtual car key | touch screen car key

digital car key mercedes | digital car key fob

Connected car moves into second gear with digital key specifications ...

telecoms.com/.../connected-car-moves-into-second-gear-with-digital-key-specification...

5 days ago - The **Car** Connectivity Consortium has announced the publication of the **Digital Key** Release 1.0 specification to allow drivers to download ...

Perfectly keyless - Bosch Mobility Solutions

https://www.bosch-mobility-solutions.com/en/products-and.../perfectly-keyless/

Both passive **vehicle** access and start are controlled by a **digital key** on a mobile phone. .... A new smartphone can be **connected** with the **vehicle** at any time,.

Images for digital key connected car

More images for digital key connected car

Report images

CCC Announces Digital Key 1.0 Standard | auto connected car news

www.autoconnectedcar.com/2018/06/ccc-announces-digital-key-1-0-standard/

5 days ago - The Release 1.0 specification provides a generalized deployment method that allows **vehicle** OEMs to securely transfer a **digital key** ...

KOREA UNIVERSITY 1905