

*Operating
Systems:
Internals
and
Design
Principles*

Chapter 15 Operating System Security

Ninth Edition
By William Stallings



Computer Security Law and Ethics



- Laws governing data trespass and data communication crimes
 - Criminal law (in Finland) http://www.laki24.fi/riri-rikokset-tieto_viestintarikokset-tietomurto/
 - “Intruding ... into system ... fine or prison max 1 year”
 - “Intruding ... using special software or device ... prison max 3 years”
 - “Just trying to intrude is punishable... “
- Univ of Helsinki IT usage and privacy policies
 - “Not allowed to ... trying to find security holes... intruding to systems... Use of information addressed or belonging to somebody else...”
<https://www.helsinki.fi/fi/it/tietojarjestelmien-kayttosaannot>
- Ethics for IT professionals <http://www.tivia.fi/julkaisut/etiikan-ohjeet>
 - “Must not misuse your position ...”
- 2 ■ “Must know laws concerning information security ...”

tietomurto, viestintärikos

Some Finnish Laws Governing Data Security

- Arkistolaki (831/1994)
- Henkilötietolaki (HetiL, 523/1999)
- Julkisuuslaki (JulkL, 621/1999)
- Laki yksityisyyden suojasta työelämässä (TETSL, 759/2004)
- Rikoslaki (39/1889, luku 35:1,2 §; luku 38:2 §, luku 38:3–4 §; luku 38:8 §)
- Suomen perustuslaki (731/1999, 10-12§)
- Sähköisen viestinnän tietosuojalaki (SVTSL, 516/2004)
- Tekijänoikeuslaki (404/1961)
- Yliopistolaki (558/2009)

3

Source: <https://www.helsinki.fi/fi/it/tietojarjestelmien-kayttosaannot>

Key Objectives of Computer Security

■ Confidentiality luottamuksellisuus

- *Data confidentiality* assures that private or confidential information is not made available or disclosed to unauthorized individuals
- *Privacy* assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed

■ Integrity koskemattomuus

- *Data integrity* assures that information and programs are changed only in a specified and authorized manner
- *System integrity* assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system

■ Availability saatavuus

- assures that systems work promptly and service to authorized users is not denied

Scope
of
System
Security

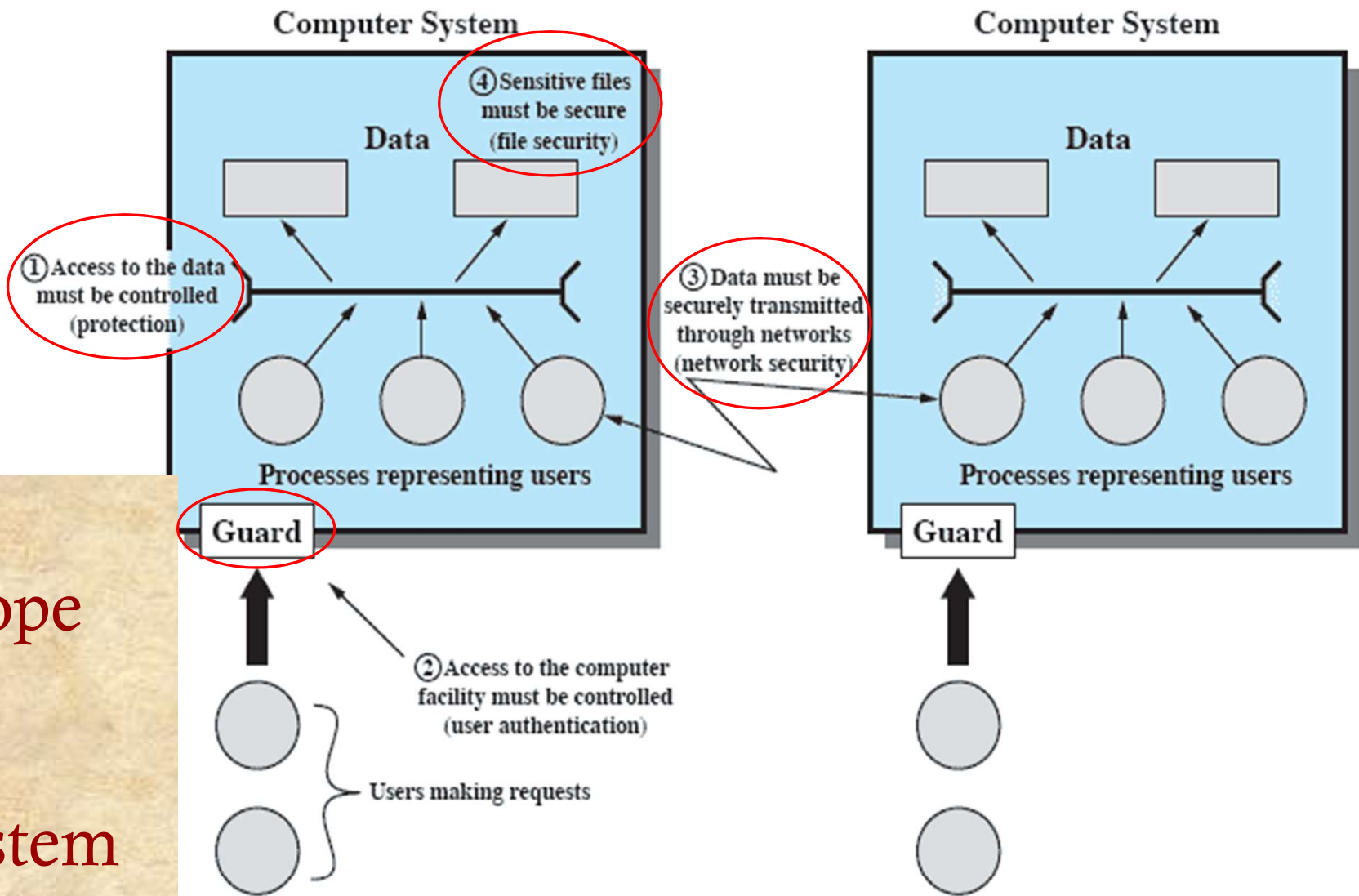
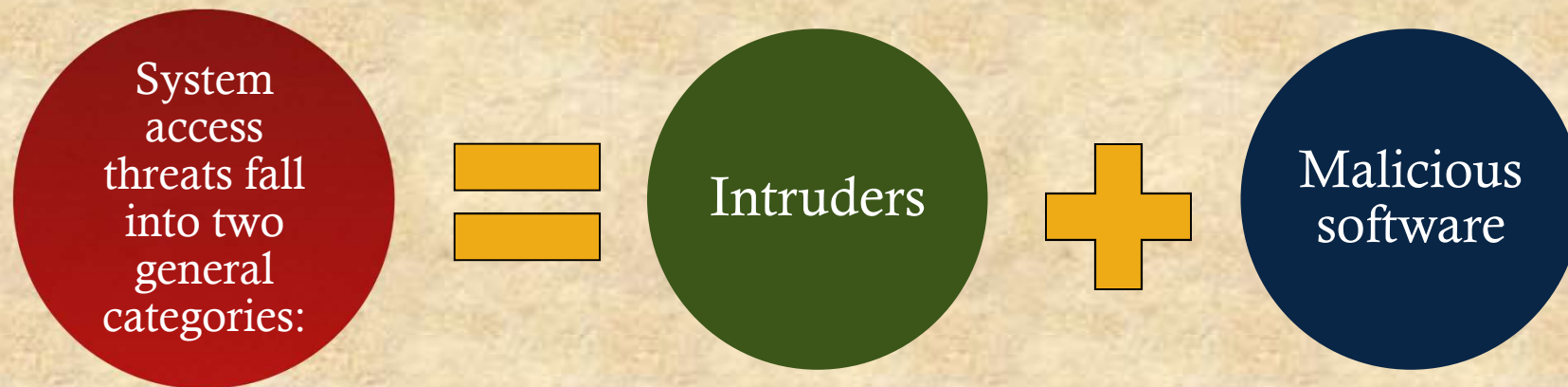


Fig 14.12 [Sta 12] Scope of System Security

System Access Threats



Threat: Intruders

Masquerader

An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account

Misfeasor

A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges

Clandestine user

An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

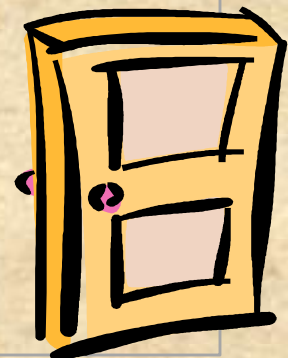
Threat: Malicious Software

- Programs that exploit vulnerabilities in computing systems
- Also referred to as malware
- Can be divided into two categories:
 - Parasitic
 - Fragments of programs that cannot exist independently of some actual application program, utility, or system program
 - Viruses, logic bombs, and backdoors are examples
 - Independent
 - Self-contained programs that can be scheduled and run by the operating system
 - Worms and bot programs are examples

Backdoor

takaovi, salaovi

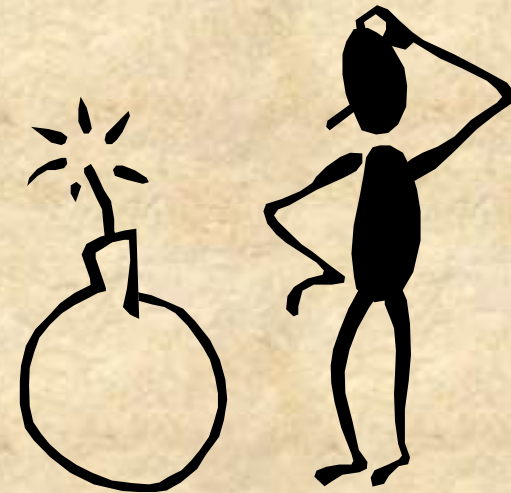
- Also known as a trapdoor
- A secret entry point into a program that allows someone to gain access without going through the usual security access procedures
- A *maintenance hook* is a backdoor that programmers use to debug and test programs
- Become threats when unscrupulous programmers use them to gain unauthorized access
- It is difficult to implement operating system controls for backdoors



Logic Bomb

logiikkapommi

- One of the oldest types of program threat
- Code embedded in some legitimate program that is set to “explode” when certain conditions are met
- Once triggered a bomb may alter or delete data or entire files, cause a machine halt, or do some other damage



Viruses

- Software that “infects” other programs by modifying them
 - carries instructional code to self duplicate
 - becomes embedded in a program on a computer
 - when the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program
 - infection can be spread by swapping disks from computer to computer or through a network
- A computer virus has three parts:
 - an infection mechanism
 - trigger
 - payload

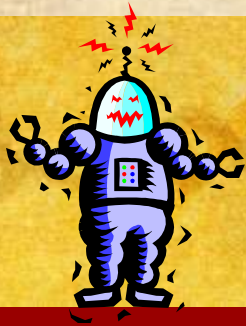




Worms

mato

- A program that can replicate itself and send copies from computer to computer across network connections
- Upon arrival the worm may be activated to replicate and propagate again
- In addition to propagation the worm usually performs some unwanted function
- Actively seeks out more machines to infect and each machine that is infected serves as an automate launching pad for attacks on other machines



Bots

piilo-ohjelma



- A program that secretly takes over another Internet-attached computer and then uses that computer to launch attacks that are difficult to trace to the bot's creator
 - also known as a Zombie or drone
- Typically planted on hundreds or thousands of computers belonging to unsuspecting third parties
- Collection of bots acting in a coordinated manner is a **botnet**
- A botnet exhibits three characteristics:
 - 1) the bot functionality
 - 2) a remote control facility
 - 3) a spreading mechanism to propagate the bots and construct the botnet

bottiverkko

Countermeasures

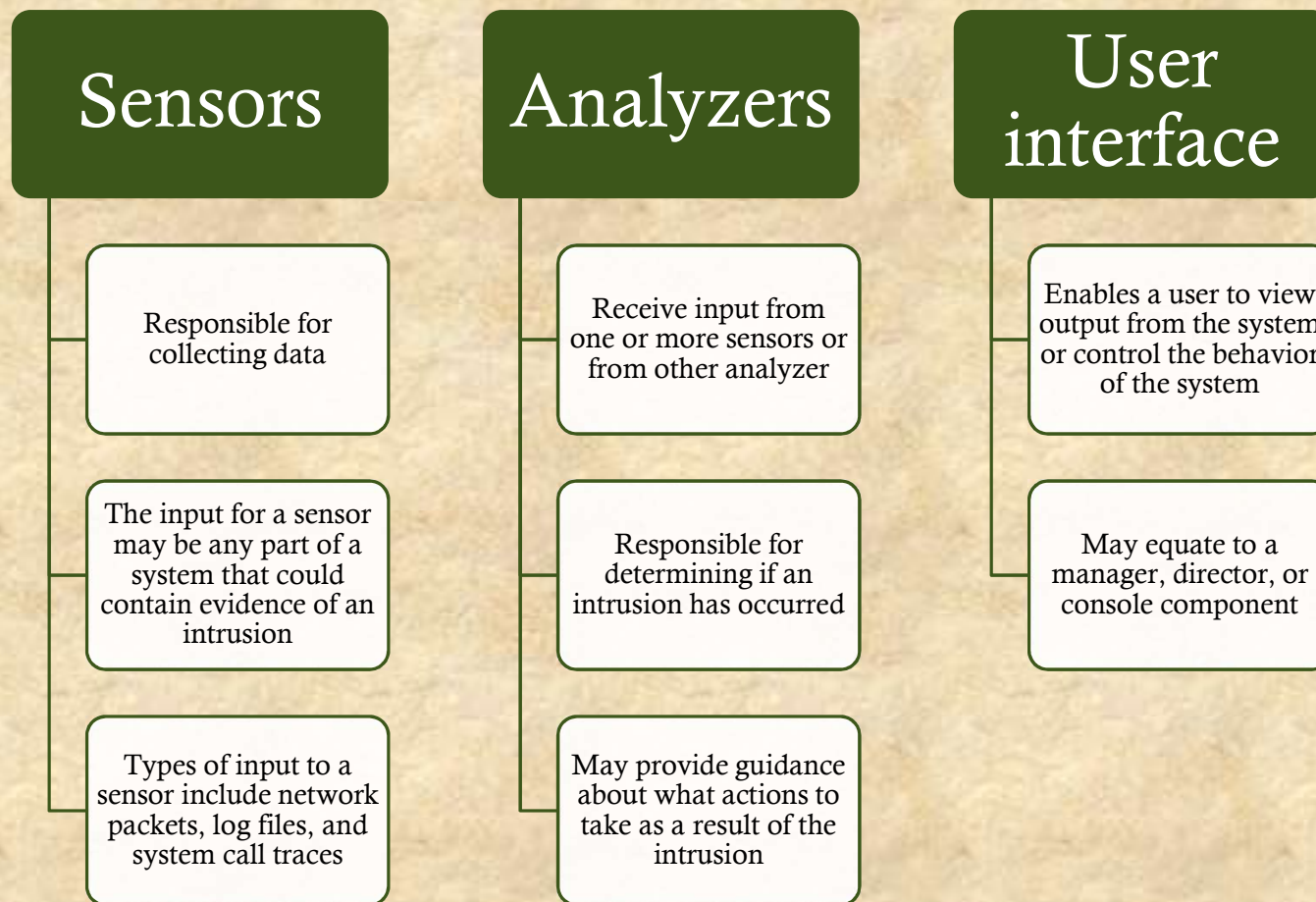
- RFC 4949 (*Internet Security Glossary*) defines intrusion detection as a security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner
- Intrusion Detection Systems (IDS) can be classified as:
 - Host-based IDS
 - Monitors the characteristics of a single host and the events occurring within that host for suspicious activity
 - Network-based IDS
 - Monitors network traffic for particular network segments or devices and analyzes network, transport, and application protocols to identify suspicious activity

Intrusion Detection - Basic Principles



- Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified
- If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised
- An effective IDS can serve as a deterrent, thus acting to prevent intrusions
- Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen intrusion prevention measures

Countermeasure: IDS Components



Countermeasure: 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

Means of Authentication

- Something the individual **knows**

5487

- Examples include a password, a personal identification number (PIN), or answers to a prearranged set of questions

- Something the individual **possesses**

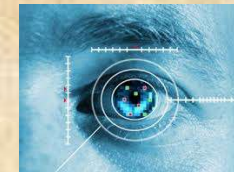


- Examples include electronic keycards, smart cards, and physical keys
- Referred to as a *token*

18

- Something the individual **is** (static biometric authentication)

- 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

click click – clicketi click click click

Access Control

- 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

Firewalls

Design goals:

- 1) All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall
- 2) Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies
- 3) The firewall itself is immune to penetration. This implies the use of a hardened system with a secured operating system. Trusted computer systems are suitable for hosting a firewall and often required in government applications

Security Issues



If a process has not declared a portion of its memory to be sharable, then no other process should have access to the contents of that portion of memory

If a process declares that a portion of memory may be shared by other designated processes then the security service of the OS must ensure that only the designated processes have access



Buffer Overflow Attacks

- Security threat related to memory management
- Also known as a buffer overrun
- Can occur when a process attempts to store data beyond the limits of a fixed-sized buffer
- One of the most prevalent and dangerous types of security attacks



Buffer Overflow Attacks

- Also known as a *buffer overrun*
- Defined in the NIST (National Institute of Standards and Technology) *Glossary of Key Information Security Terms* as:

“A condition at an interface under which more input can be placed into a buffer or data-holding area than the capacity allocated, overwriting other information. Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system”
- One of the most prevalent and dangerous types of security attacks

Buffer Overflow Example

```
int main(int argc, char *argv[]) {  
    int valid = FALSE;  
    char str1[8];  
    char str2[8];  
  
    next_tag(str1);  
    gets(str2);  
    if (strncmp(str1, str2, 8) == 0)  
        valid = TRUE;  
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);  
}
```

Set correct comparison string to str1: START

No check of length of str2,
may modify also str1 (in activation record).

What if return address is modified?

What if code inserted, and jumped into?

(a) Basic buffer overflow C code

```
$ cc -g -o buffer1 buffer1.c  
$ ./buffer1  
START  
buffer1: str1(START), str2(START), valid(1) ok  
$ ./buffer1  
EVILINPUTVALUE  
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)  
$ ./buffer1  
BADINPUTBADINPUT  
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

ok result, but
str1 corrupted

erroneous result,
str1 corrupted

24

(b) Basic buffer overflow example runs

Fig. 15.1

Buffer Overflow Stack Values

How would you fix the problem?

Fig 15.2

Memory Address	Before gets (str2)	After gets (str2)	Contains Value of
...	
bffffbf4	34fcffbf 4 . . .	34fcffbf 3 . . .	<u>argv</u>
bffffbf0	01000000	01000000	<u>argc</u>
<u>bffffbec</u>	c6bd0340 . . . @	c6bd0340 . . . @	return <u>addr</u>
bffffbe8	08fcffbf	08fcffbf	old base <u>ptr</u>
bffffbe4	00000000	01000000	valid
bffffbe0	80640140 . d . @	00640140 . d . @	
<u>bffffbdc</u>	54001540 <u>T . . @</u>	4e505554 N P U T	<u>str1[4-7]</u>
bffffbd8	53544152 S T A R	42414449 B A D I	<u>str1[0-3]</u>
bffffbd4	00850408	4e505554 N P U T	<u>str2[4-7]</u>
bffffbd0	30561540 O V . @	42414449 B A D I	<u>str2[0-3]</u>
...	

Exploiting Buffer Overflow

- To exploit any type of buffer overflow the attacker needs:
 - To identify a buffer overflow vulnerability in some program that can be triggered using externally sourced data under the attackers control
 - To understand how that buffer will be stored in the processes memory, and hence the potential for corrupting adjacent memory locations and potentially altering the flow of execution of the program

Defending Against Buffer Overflows

- Prevention E.g., do not use `gets()` `fgets(str2, 8, stdin)`
- Detecting and aborting E.g., save token names as read only constants
- Countermeasure categories:

Compile-time Defenses

- aim to harden programs to resist attacks in new programs

Run-time Defenses

- aim to detect and abort attacks in existing programs

Automatic check for index overflow code, etc.

Keep literals in read-only segment. Disallow code execution in stack or heap. Mark all code non-executable by default. Etc.

Intel Management Engine Buffer Overflow (etc) Bugs

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

- Intel Management Engine
 - Extra processor (core) running management OS (MINIX) code
 - Exists in HW/OS outside std OS, normal updates not possible
- Multiple buffer overflow (and other) bugs found in 2017
 - “allows attacker with local access to execute arbitrary code”
 - “allows attacker with local access to execute arbitrary code with AMT (Active Mgt Techn) execution privilege.”
 - “allows attacker with remote Admin access to execute arbitrary code with AMT execution privilege”
- Updates must be distributed via hardware vendors (Lenovo, etc)

http://www.theregister.co.uk/2017/11/20/intel_flags_firmware_flaws/

30 <https://security-center.intel.com/advisory.aspx?intelid=INTEL-SA-00086&languageid=en-fr>

File System Access Control

- Identifies a user to the system
- 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 System) Security with Access Matrix

pääsymatriisi

- The basic elements are:
 - **subject** – an entity capable of accessing objects
 - **object** – anything to which access is controlled
 - **access rights** – the way in which an object is accessed by a subject

object

Fig. 15.3

	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	access rights	Own R W		Inquiry Debit

subject

(a) Access matrix

Access Control Lists

- Per object
- A matrix may be decomposed by access matrix columns, yielding **access control lists**
- The access control list lists users and their permitted access rights

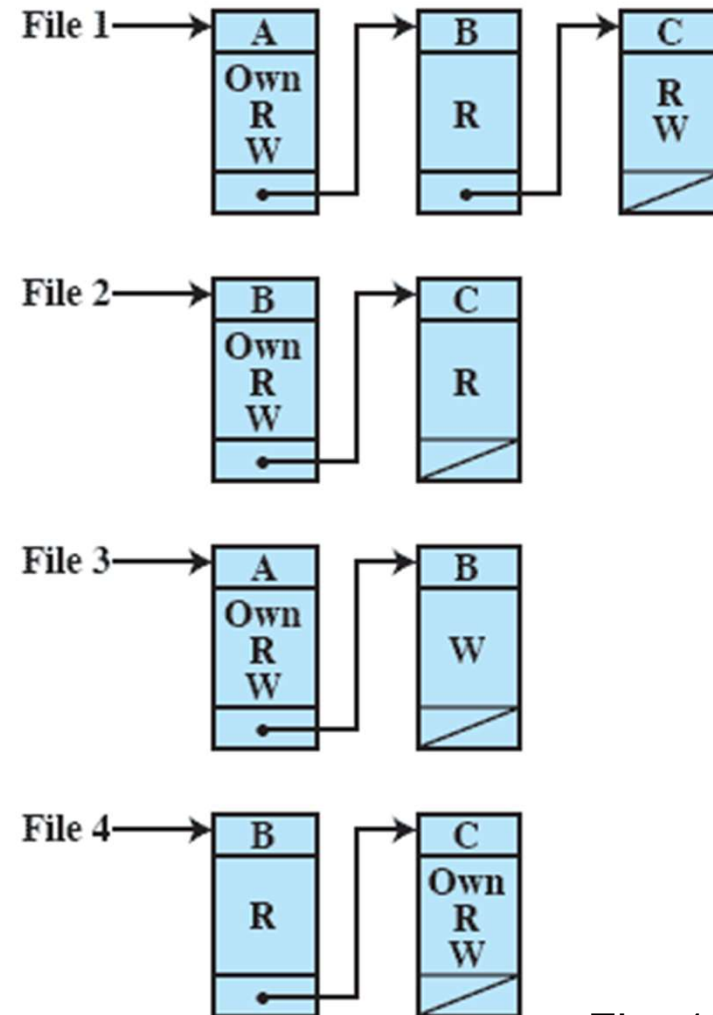


Fig. 15.3

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

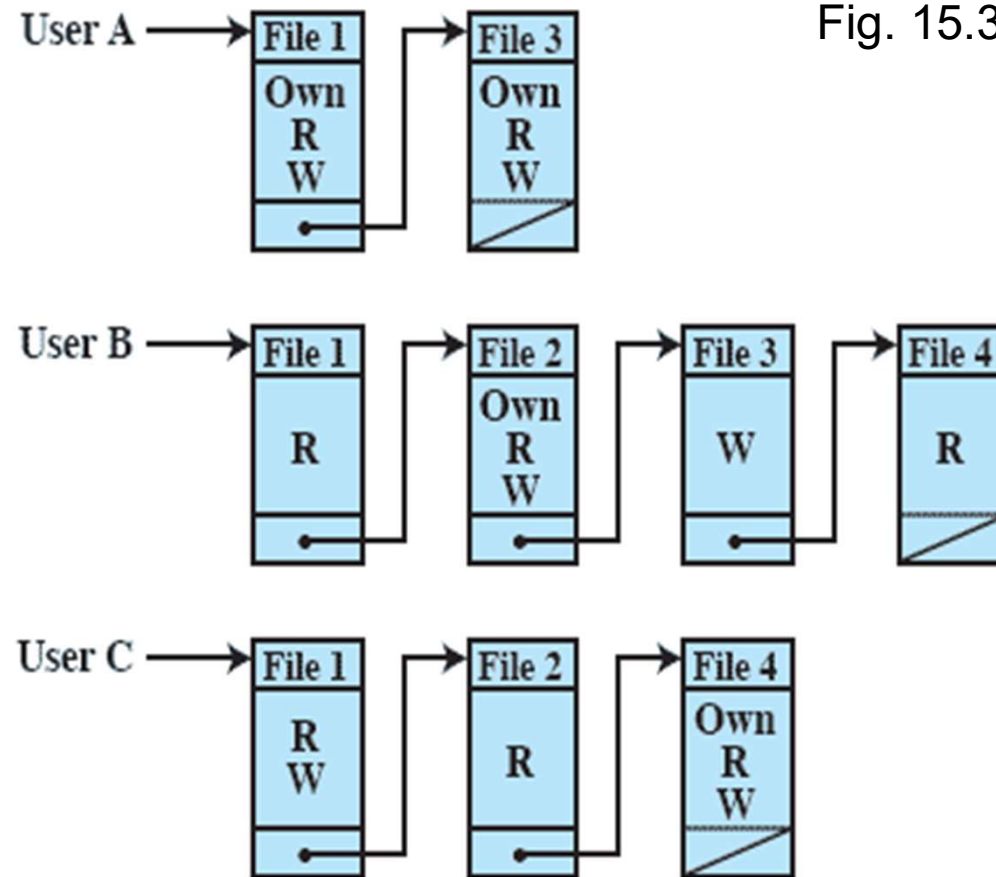
Capability Lists

- Per user
- Decomposition by rows yields **capability tickets** valtakirja
- A **capability ticket** specifies authorized objects and operations for a user
- Unforgeable
 - Encryption
 - System memory

35

valtakirjalista

Fig. 15.3



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

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) **harkinnanvarainen**
 - Controls access based on the identity of the requestor and on access rules stating what requestors are allowed to do
 - Mandatory access control (MAC) **pakollinen**
 - Controls access based on comparing security labels with security clearances
 - Role-based access control (RBAC) **rooliperustainen**
 - 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) **attribuuttiperustainen**
 - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions

Extended Access Control Matrix

		OBJECTS								
		subjects			files		processes		disk drives	
		S ₁	S ₂	S ₃	F ₁	F ₂	P ₁	P ₂	D ₁	D ₂
SUBJECTS	S ₁	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	S ₂		control		write *	execute			owner	seek *
	S ₃			control		write	stop			

* - copy flag set

Figure 15.4 Extended Access Control Matrix

Access Rights



- ***None***
 - the user would not be allowed to read the user directory for the file
- ***Knowledge***
 - the user can determine that the file exists and who its owner is and can then petition the owner for additional access rights
- ***Execution***
 - the user can load and execute a program but cannot copy it
- ***Reading***
 - the user can read the file for any purpose, including copying and execution
- ***Appending***
 - the user can add data to the file but cannot modify or delete any of the file's contents
- ***Updating***
 - the user can modify, delete, and add to the file's data
- ***Changing protection***
 - the user can change the access rights granted to other users
- ***Deletion***
 - the user can delete the file from the file system

User Access Rights

Owner

usually the
initial creator
of the file

has full rights

may grant
rights to
others

Specific Users

individual
users who are
designated by
user ID

User Groups

a set of users
who are not
individually
defined

All

all users who
have access to
this system

these are
public files
(if path is
public)

Organization of the Access Control Function

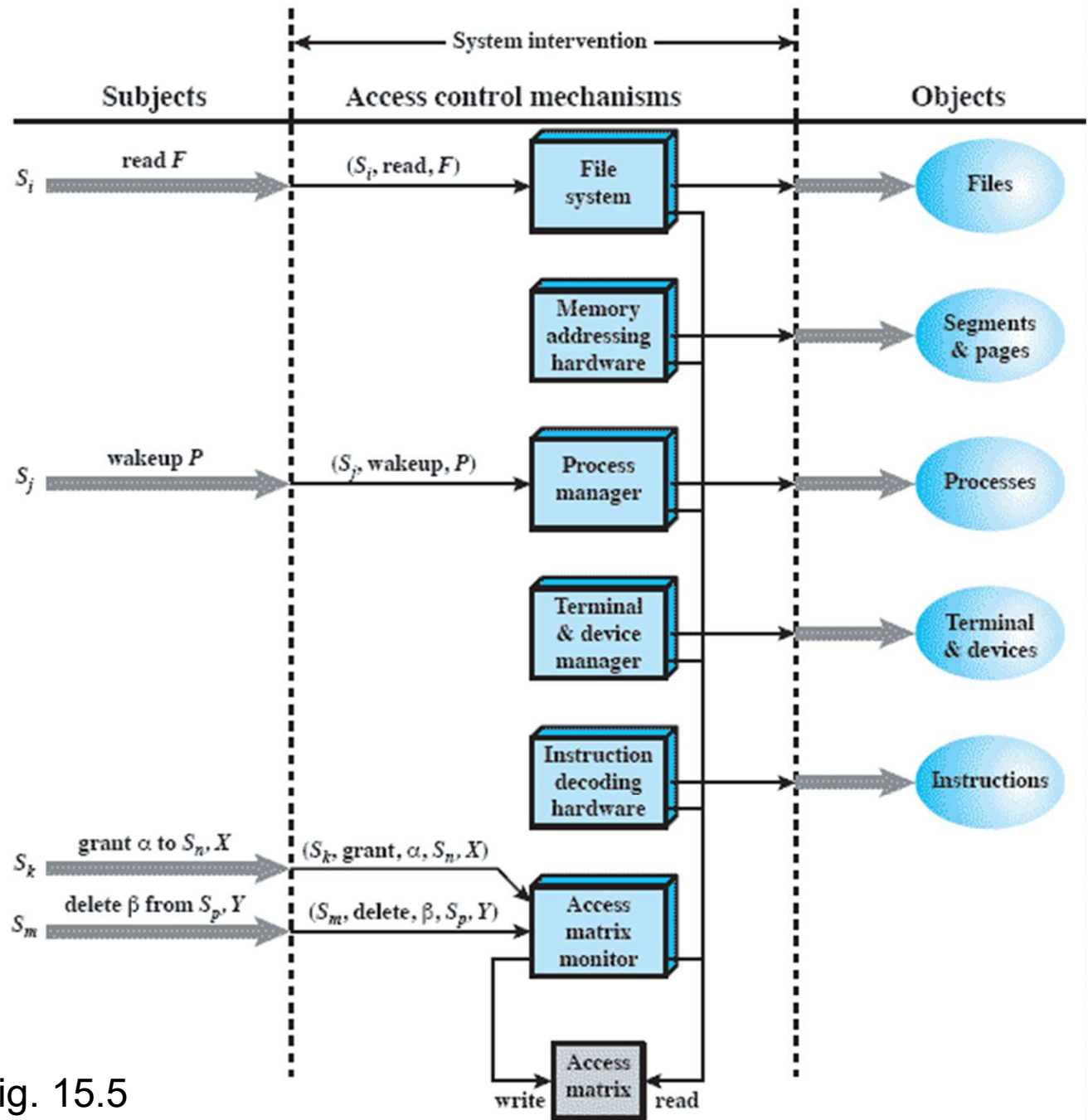


Fig. 15.5

Access Control System Commands

Tbl 15.1

Rule	Command (by S_o)	Authorization	Operation
R1	transfer $\left\{ \begin{matrix} \alpha^* \\ \alpha \end{matrix} \right\}$ to S, X	copy access right ' α^* ' in $A[S_o, X]$	store $\left\{ \begin{matrix} \alpha^* \\ \alpha \end{matrix} \right\}$ in $A[S, X]$
R2	grant $\left\{ \begin{matrix} \alpha^* \\ \alpha \end{matrix} \right\}$ to S, X	'owner' in $A[S_o, X]$	store $\left\{ \begin{matrix} \alpha^* \\ \alpha \end{matrix} \right\}$ in $A[S, X]$
R3	delete α from S, X	'control' in $A[S_o, S]$ or 'owner' in $A[S_o, X]$	delete α from $A[S, X]$
R4	$w \leftarrow$ read S, X	'control' in $A[S_o, S]$ or 'owner' in $A[S_o, X]$	copy $A[S, X]$ into w
R5	create object X	None	add column for X to A ; store 'owner' in $A[S_o, X]$
R6	destroy object X	'owner' in $A[S_o, X]$	delete column for X from A
R7	create subject S	none	add row for S to A ; execute create object S ; store 'control' in $A[S, S]$
R8	destroy subject S	'owner' in $A[S_o, S]$	delete row for S from A ; execute destroy object S

Role-Based Access Control

- Based on the roles that users assume in a system rather than the user's identity
- Models define a role as a job function within an organization
- Systems assign access rights to roles instead of individual users
 - in turn, users are assigned to different roles, either statically or dynamically, according to their responsibilities
- NIST standard, that requires support for access control and administration through roles

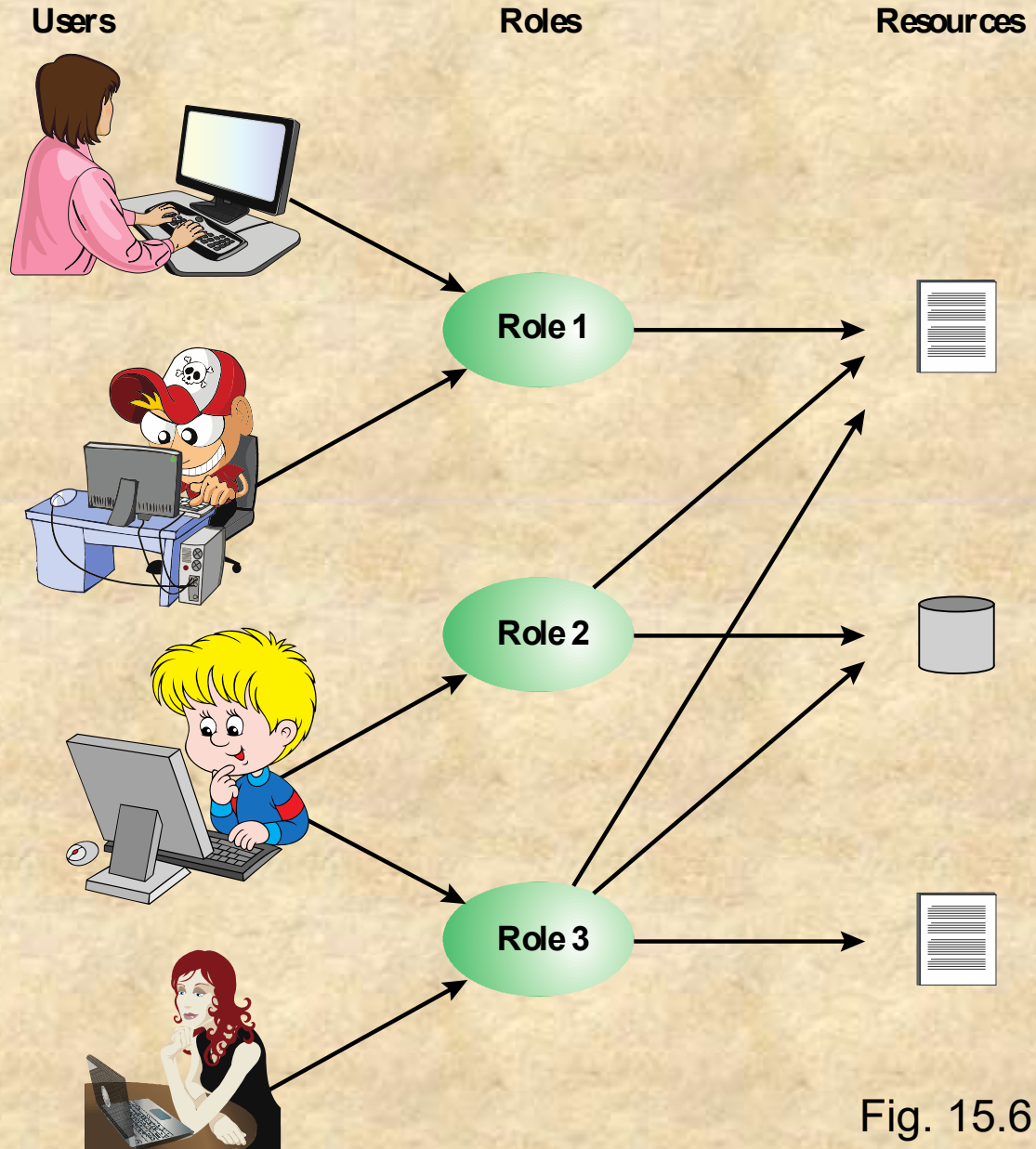
NIST = national institute of standards & technology (USA)



Users

Roles

Resources



Access Control Matrix Representation of Role Based Access Control

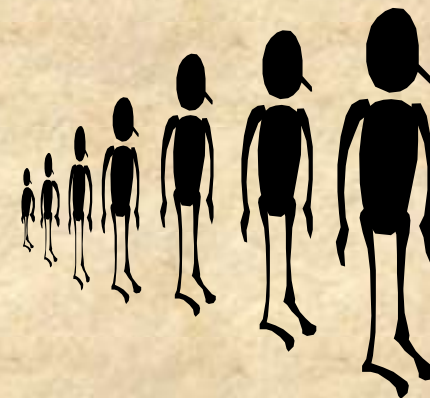
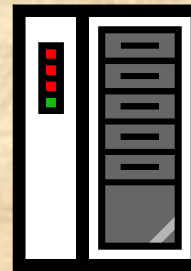
	R ₁	R ₂	...	R _n
U ₁	×			
U ₂	×			
U ₃		×		×
U ₄				×
U ₅				×
U ₆				×
⋮				
U _m	×			

Fig. 15.7

		OBJECTS								
		R ₁	R ₂	R _n	F ₁	F ₁	P ₁	P ₂	D ₁	D ₂
ROLES	R ₁	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	R ₂		control		write *	execute			owner	seek *
	⋮									
	R _n			control		write	stop			

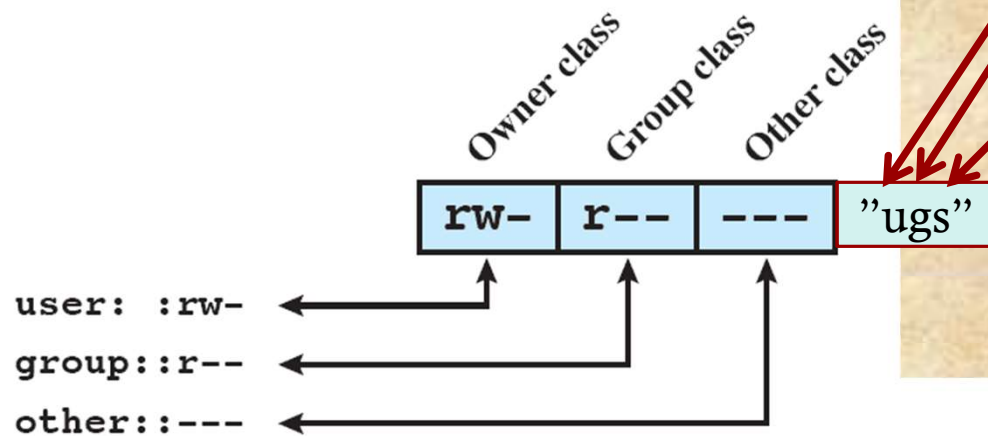
Access Control Lists in UNIX

- FreeBSD allows the administrator to assign a list of UNIX user IDs and groups to a file
- Any number of users and groups can be associated with a file, each with three protection bits (read, write, execute)
- A file may be protected solely by the traditional UNIX file access mechanism
- FreeBSD files include an additional protection bit that indicates whether the file has an extended ACL



UNIX File Access Control

Fig. 15.8

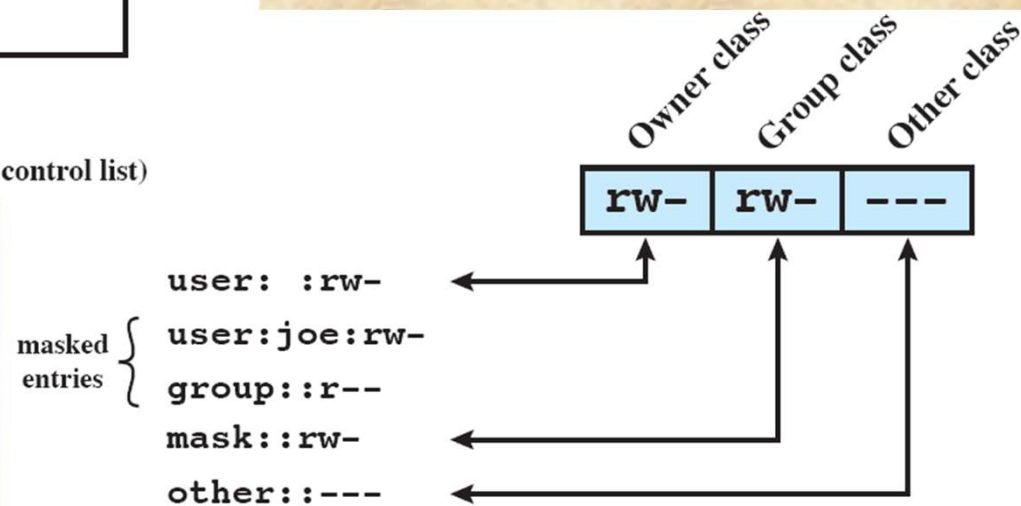


SetUID – effective user ID (owner)
 SetGID – effective group ID
 "Sticky" – rename, move, delete

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



46



(b) Extended access control list

Discuss

1.11.2019

Windows 7 Security

- Exploits object-oriented concepts to provide a powerful and flexible access control capability
- Provides a uniform access control facility that applies to processes, threads, files, semaphores, windows, and other objects
- Access control is governed by:
 - Access token associated with each process
 - Process can impersonate another process
 - Use some other access token for a while
 - E.g., a server can impersonate a client (by using clients access token)
 - Inherited by spawned processes
 - Security descriptor associated with each object for which interprocess access is possible

Windows Access Control Scheme

- When a user logs on to a Windows system a name/password scheme is used to authenticate the user
- If the logon is accepted a process is created for the user and an access token is associated with that process object
 - The access token includes a security ID (SID) which is the identifier by which this user is known to the the system for purposes of security
 - The token also contains SIDs for the security groups to which the user belongs

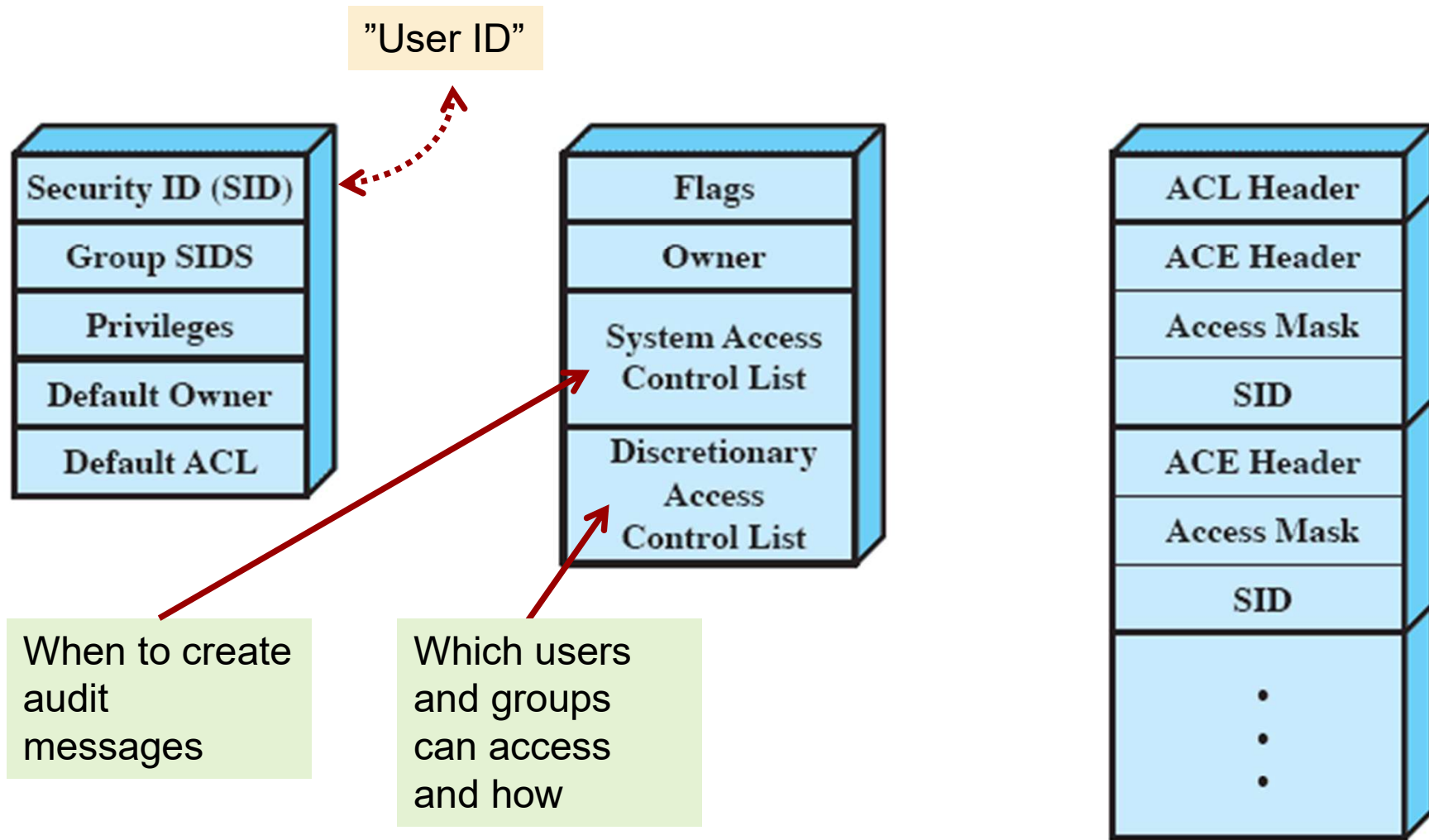
The access token

It keeps all necessary security information together to speed access validation

serves two purposes:

It allows each process to modify its security characteristics in limited ways without affecting other processes running on behalf of the user

Windows Access Token



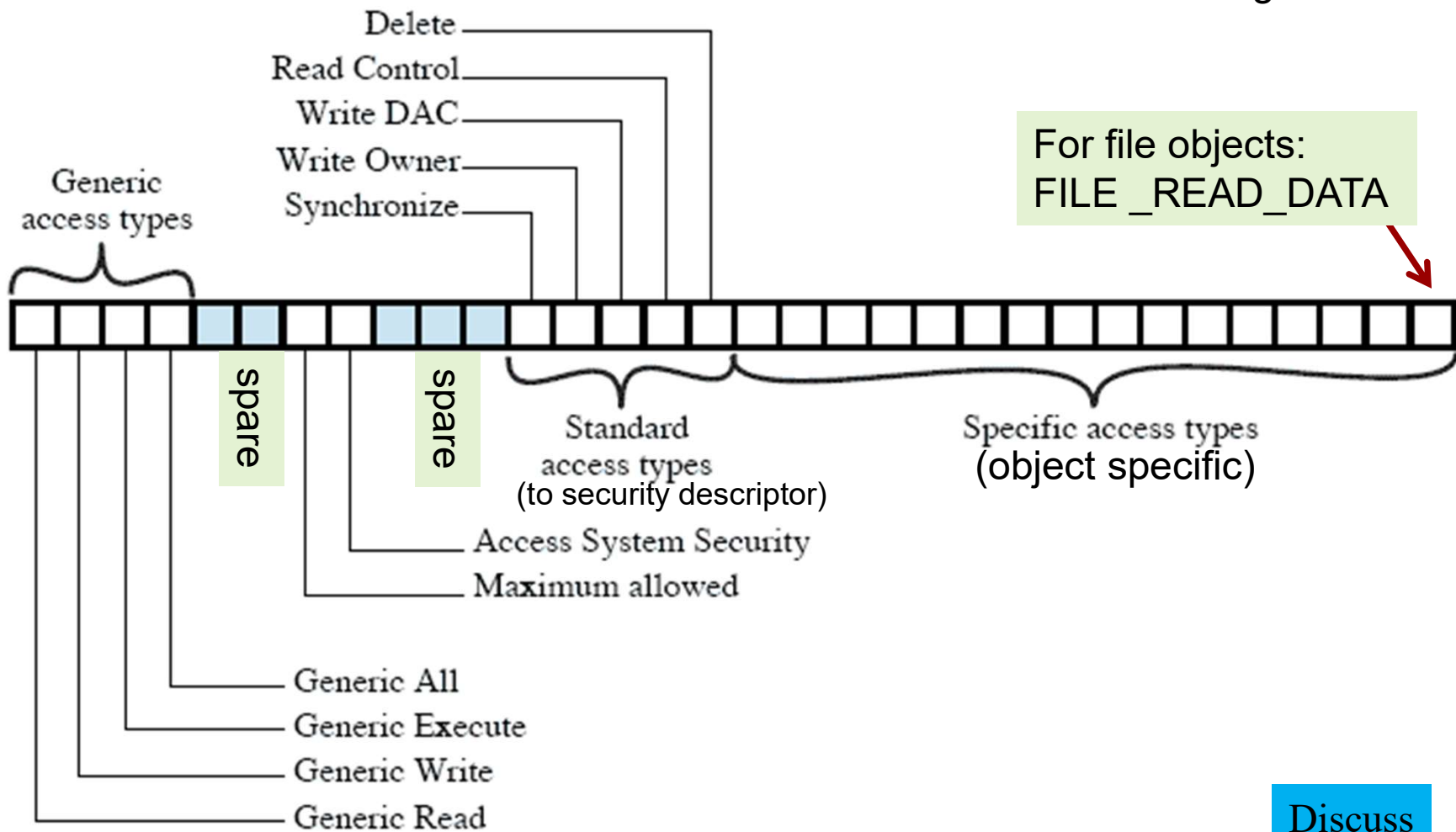
(a) Access token

(b) Security descriptor

(c) Access control list

Windows Access Mask

Fig. 15.10




Operating Systems Hardening

- Basic steps to use to secure an operating system:
 - Install and patch the operating system
 - Harden and configure the operating system to adequately address the identified security needs of the system by:
 - Removing unnecessary services, applications, and protocols
 - Configuring users, groups and permissions
 - Configuring resource controls
 - Install and configure additional security controls, such as antivirus, host-based firewalls, and intrusion detection systems (IDS), if needed
 - Test the security of the basic operating system to ensure that the steps taken adequately address its security needs


<https://www.asd.gov.au/infosec/top-mitigations/top-4-strategies-explained.htm>

Operating System Installation: Initial Setup and Patching

System security begins with the installation of the operating system



Ideally new systems should be constructed on a protected network



The initial installation should comprise the minimum necessary for the desired system, with additional software packages included only if they are required for the function of the system



The overall boot process must also be secured



Care is also required with the selection and installation of any additional device driver code, since this executes with full kernel level privileges, but is often supplied by a third party

Security Maintenance

- The process of security maintenance includes the following steps:

Performing regular backups

Monitoring and analyzing logging information

Regularly testing system security

Recovering from security compromises

Using appropriate software maintenance processes (patch and update all critical software, monitor and revise configuration as needed)

Logging

- Effective logging helps ensure that in the event of a system breach or failure, system administrators can more quickly and accurately identify what happened and more effectively focus their remediation and recovery efforts
- Logging information can be generated by the system, network, and applications
- The range of logging data acquired should be determined during the system planning stage
- Logging can generate significant volumes of information so it is important that sufficient space is allocated for them
- A suitable automatic log rotation and archive system should be configured to assist in managing the overall size of the logging information
- Some form of automated analysis is preferred as it is more likely to identify abnormal activity
 - Manual analysis of logs is tedious and is not a reliable means of detecting adverse events

Data Backup and Archive

- Performing regular backups of data on a system is another critical control that assists with maintaining the integrity of the system and user data
- The needs and policy relating to backup and archive should be determined during the system planning stage
 - Key decisions include whether the copies should be kept online or offline and whether copies should be stored locally or transported to a remote site
- Backup
 - The process of making copies of data at regular intervals, allowing the recovery of lost or corrupted data over relatively short time periods of a few hours to some weeks
- Archive
 - The process of retaining copies of data over extended periods of time, being months or years, in order to meet legal and operational requirements to access past data

Summary

- Intruders and malicious software
 - System access threats
 - Countermeasures
- Buffer overflow
 - Buffer overflow attacks
 - Compile time defences
 - Runtime defenses
- Access control
 - File system access control
 - Access control policies
- UNIX access control
- Windows access control
- Operating systems hardening
- Security maintenance