

# **SC3010**

# **Computer Security**

## **Lecture 5: Operating System Security (I)**

# Security Challenges in Modern OS

---

## From single-user to multi-user

- ▶ DOS is truly single user
- ▶ MacOS, Linux, NT-based Windows are multi-user
- ▶ Cloud computing allows multiple users all over the world to run on the same system, and they do not know each other.
- ▶ **Not all users are trusted!**

## From trusted apps to untrusted apps

- ▶ Simple real-time systems: only run one specific app from trusted sources
- ▶ Modern PCs and smartphones: run apps from third-party developers
- ▶ **Not all apps are trusted!**

## From standalone systems to networked systems

- ▶ Isolated computer systems only need to protect against physical threats.
- ▶ Once connected to networks, the system faces external unknown threats.
- ▶ **Not all network components are trusted!**

# Outline

---

- ▶ **Security Protection Stages in OS**
  - ▶ Authentication
  - ▶ Authorization with Access Control
  - ▶ Logging, Monitoring & Auditing
- ▶ **Privilege Management in OS**

# Outline

---

- ▶ **Security Protection Stages in OS**

- ▶ Authentication
- ▶ Authorization with Access Control
- ▶ Logging, Monitoring & Auditing

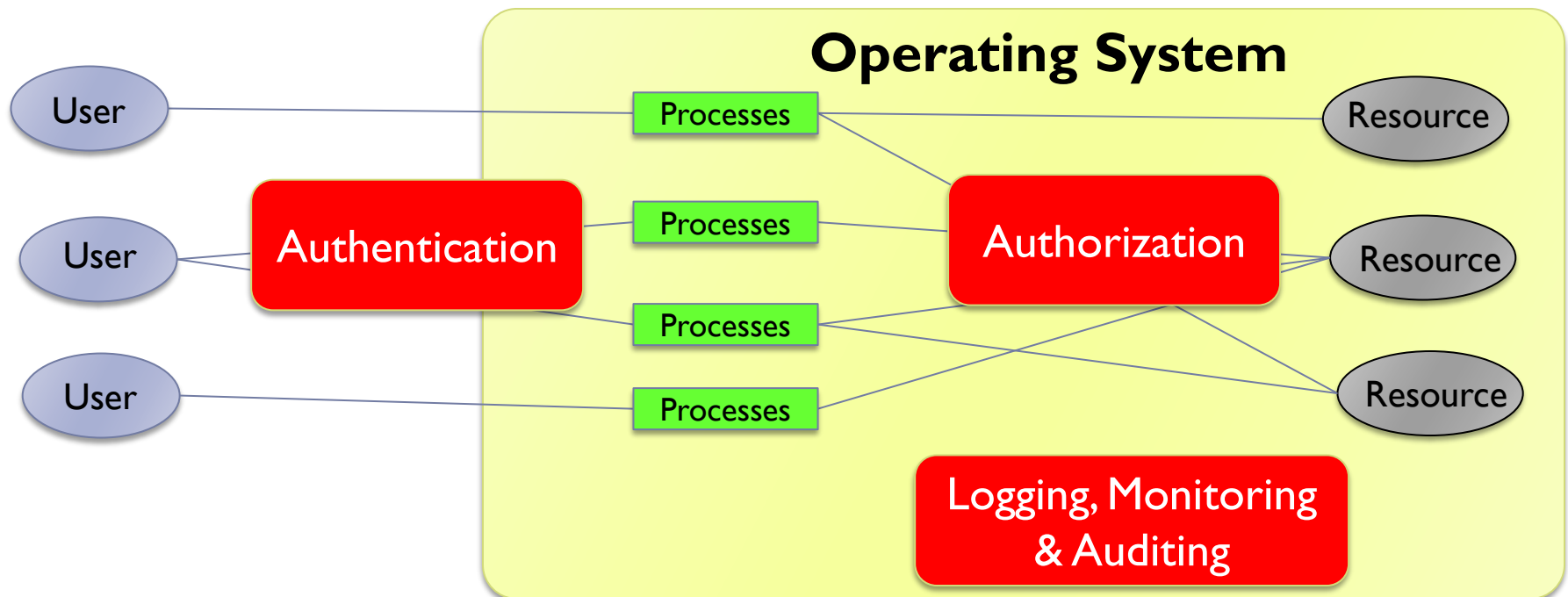
- ▶ **Privilege Management in OS**

# Security Protection from OS

OS is responsible for protecting the apps and resources inside it.

- ▶ OS controls what users/processes can do
- ▶ OS prevents what users/processes cannot do







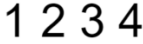


## Protection Stages



# Authentication

## How does a computer know if I am a correct user?

- ▶ **Something you know:** password, PIN, public/private keys...
- ▶ **Something you have:** smartcard, hardware tokens...
- ▶ **Something you are:** biometrics, face recognition, voice recognition...

Knowledge Factor (something you know)	Possession Factor (something you have)	Inherence Factor (something you are)
 Password	 Smartphone	 Fingerprint
 Security Question	 Smart Card	 Retina Pattern
 PIN	 Hardware Token	 Face Recognition

# Something You Know: Password

## Password is the most common way to prove who you are

- ▶ Adopted by various networking websites and applications
- ▶ The security of the password-based authentication mechanism depends on the strength of the selected password, i.e., the chance attacker can guess the password.
- ▶ The trade-off between the password security and convenience:
  - Weak password is easy to memorize, but also easy to be guessed.
  - Complex password is strong but results in frustrated users



Nanyang Technological University

Sign in with your organizational account

Sign in

Sign in using your network account e.g

- username@staff.main.ntu.edu.sg
- username@student.main.ntu.edu.sg
- username@assoc.main.ntu.edu.sg
- username@niestaff.cluster.nie.edu.sg
- username@niestudent.cluster.nie.edu.sg

# Weak Password

A weak password is a character combination that is easy for friends, bad actors or password-hacking software to guess

- ▶ Short passwords: a single word (e.g., password) or numerical phrase (e.g., 12345).
- ▶ Recognizable keystroke patterns: take a look at your keyboard and find QWERTY
- ▶ Personal information in passwords: e.g., date of birth, address, name
- ▶ Repeated letters or numbers: e.g., 55555, aaaa

Password Popularity – Top 20

Rank	Password	Number of Users with Password (absolute)
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	61958
5	iloveyou	51622
6	princess	35231
7	rockyou	22588
8	1234567	21726
9	12345678	20553
10	abc123	17542

Rank	Password	Number of Users with Password (absolute)
11	Nicole	17168
12	Daniel	16409
13	babygirl	16094
14	monkey	15294
15	Jessica	15162
16	Lovely	14950
17	michael	14898
18	Ashley	14329
19	654321	13984
20	Qwerty	13856



# Strong Password

---

A strong password is a long combination of unique characters that is difficult for other people to guess or technology to crack

- ▶ Lengthy combinations: long passwords with various character types, such as numbers, letters and symbols, e.g., N0r+Hc^R0|in^99
- ▶ Mnemonic: create passwords inspired by events only notable to you, e.g., a string of first letters of a meaningful sentence
- ▶ Non-dictionary words: dictionary words — formal or slang — are publicly known combinations of characters stored in a database that cybercriminals access using software to input thousands of passwords per second

A strong authentication system requires users to change their password periodically (e.g., every six months), and the new passwords must be different from the used ones.

# Something You Have

## Different types of possessions for authentication

- ▶ Tokens
- ▶ Smartcards: a physical card + a smart card reader



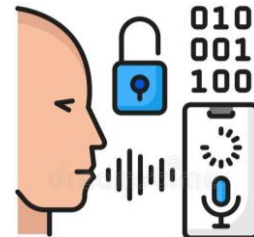
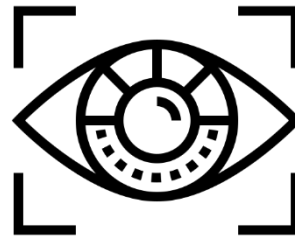
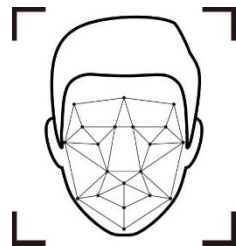
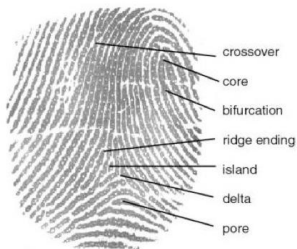
## Limitations of physical belongings

- ▶ Easy to get lost. Therefore, it is safer to combine users' knowledge with physical belongings. This is referred to as **two-factor authentication**
- ▶ High cost (e.g., \$15-\$25, banks with million customers).
- ▶ Possible to get damaged (e.g., card in the washing machine, battery death)
- ▶ Non-standard algorithms.

# Something You Are

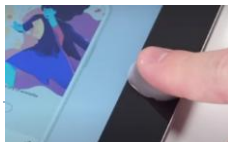
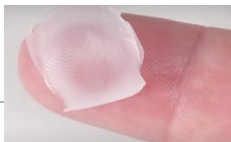
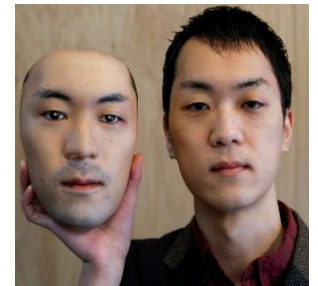
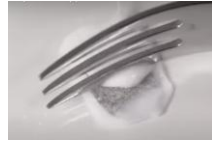
## Biometrics measure some physical characteristic

- ▶ Fingerprint, face recognition, retina scanners, voice, etc.
- ▶ Can be extremely accurate and fast



## Limitations of biometrics

- ▶ Private, but not secret. Maybe encoded on your glass, door handle
- ▶ Revocation is difficult: Sorry, your iris has been compromised, please create a new one...



# Authorization

---

## Access control

- ▶ Implement a **security policy** that specifies who or what may have access to each specific resource in a computer system, and the type of access that is permitted in each instance.
- ▶ It mediates between a user (or a process executing on behalf of a user) and system resources (e.g., applications, network sockets, firewalls).

## Three basic elements in a security policy:

- ▶ Subject: process or users
- ▶ Object: resource that is security-sensitive
- ▶ Operations: actions taken using that resource

# Subject

---

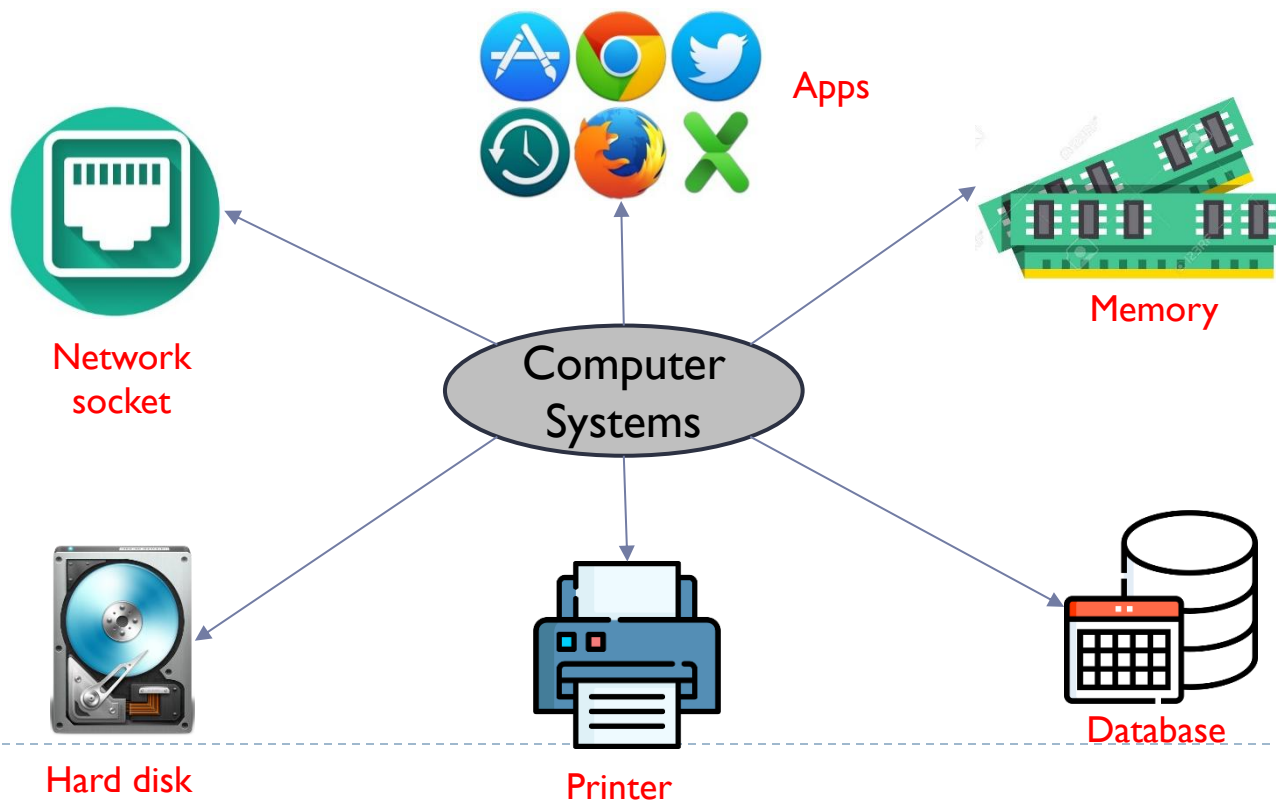
A **subject** is typically held accountable for the actions they have initiated. There can be three types of subjects.

- ▶ **Owner**: this may be the creator of a resource. For system resources, ownership may belong to a system administrator.
- ▶ **Group**: in addition to individual users, privileges can also be assigned to a group of users. A user joining the group will automatically have the corresponding privileges, while a user quitting the group will lose the corresponding permissions. A user may belong to multiple groups. The concept of groups makes it easier to manage and update the permissions.
- ▶ **Other**: the **least amount of access** is granted to users who are able to access the system but are not included in the categories of owner and group for this resource.

# Object

An **object** is a resource to which access is controlled.

- ▶ An entity used to contain and/or receive information.
- ▶ Examples: records, blocks, pages, segments, files, portions of files, directories, directory trees, mailboxes, messages, and programs.



# Operation

---

## Describes the way in which a subject may access an object

- ▶ **Read:** user may **view** information in a system resource (e.g., a file, selected records in a file, selected fields within a record, or some combination).  
Read access includes **the ability to copy or print**.
- ▶ **Write:** user may **modify** data in the system resource (e.g., files, records, programs).
- ▶ **Execute:** user may **execute** specified programs.
- ▶ **Delete:** user may **delete** certain system resources, such as files or records.
- ▶ **Create:** user may **create** new files, records, or fields.
- ▶ **Search:** user may **list** the files in a directory or otherwise **search** the directory.

# Access Control Matrix

## A popular implementation of access control policy.

- ▶ One dimension consists of identified subjects that may attempt access to the resources
- ▶ The other dimension lists the objects that may be accessed
- ▶ Each entry in the matrix indicates the access rights of a particular subject for a particular object

		Objects			
Subjects		File 1	File 2	File 3	File 4
	User A	Read Write Execute		Read Write Execute	
	User B	Read	Read Write Execute	Write	Read
	User C	Read Write	Read		Read Write Execute



# Update Access Control Matrix

## Possible changes over Access Control Matrix

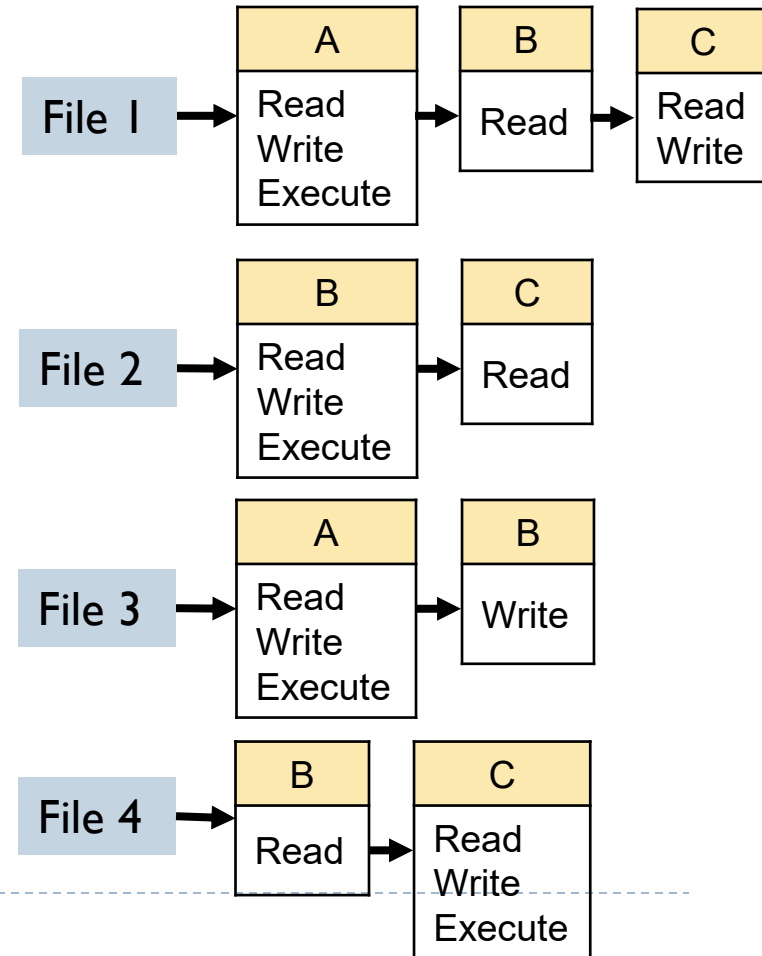
- ▶ Create subject  $s$ : add a new row  $s$ . This is typically done by the system administrator.
- ▶ Create object  $o$ : create a new column  $o$ . This is typically done by the system administrator.
- ▶ Grant permission  $r$  for subject  $s$  over object  $o$ : enter  $r$  to entry  $M_{s,o}$ . This is typically done by the resource owner or system administrator.
- ▶ Revoke permission  $r$  for subject  $s$  over object  $o$ : delete  $r$  from entry  $M_{s,o}$ . This is typically done by the resource owner or system administrator.
- ▶ Destroy subject  $s$ : delete the row  $s$ . This is typically done by the system administrator.
- ▶ Destroy object  $o$ : deletes the column  $o$ . This is typically done by the system administrator.

# Access Control List (ACL)

In practice, an access control matrix is usually sparse and can be implemented by decomposition in one of two ways

## Decomposition by columns

- ▶ For each object, ACL lists users their permitted access rights.
- ▶ ACL is convenient when determining which subjects have which access to a particular resource.

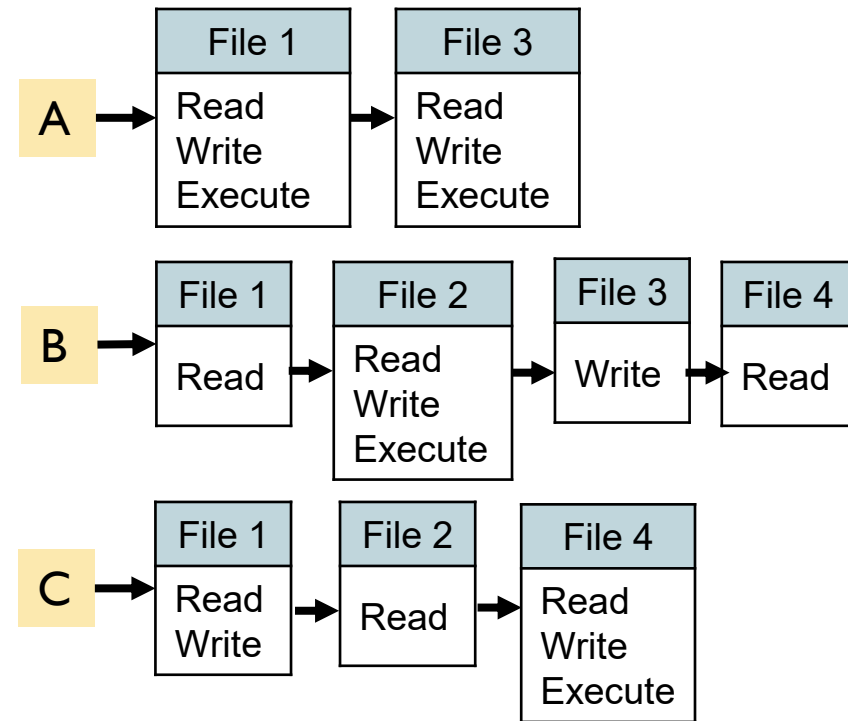


# Capability List (C-List)

In practice, an access control matrix is usually sparse and can be implemented by decomposition in one of two ways

## Decomposition by rows

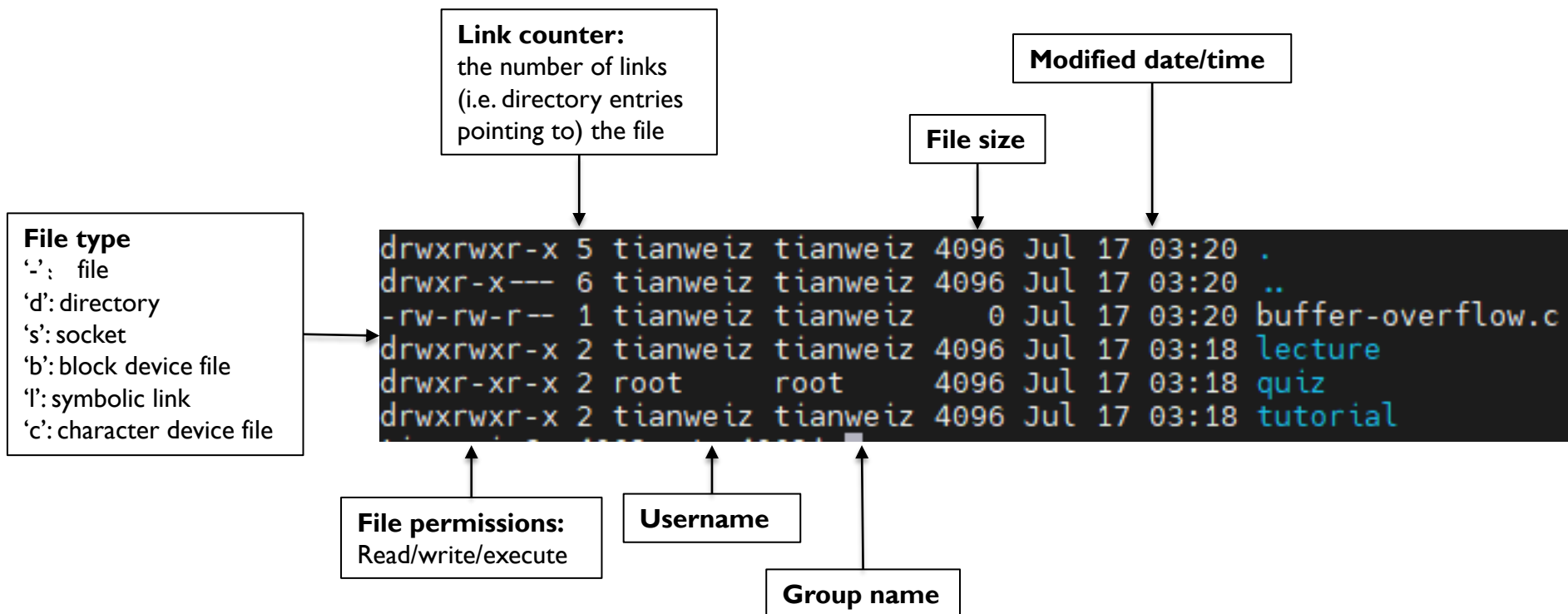
- ▶ C-list specifies authorized objects and operations for a particular user.
- ▶ C-List is convenient when determining the access rights available to a specific user.



# Example: Resource Management in Unix OS

Files, directories, memory devices, I/O devices are uniformly treated as **resources**

- ▶ These resources are the objects of access control.
- ▶ Each resource has a single user owner and group owner



# Permission Representation

---

## Three permissions with three subjects

- ▶ Read, Write, Execute
- ▶ Owner, Group, Other
- ▶ Examples:
  - `rw-r--r--`: read and write access for owner, read access for group and other.
  - `rwX-----`: read, write, and execute access for owner, no rights to group and other.

## Octal Representation

- ▶ `rw-r--r--`: `110 100 100`: `644`
- ▶ `rwX-----`: `111 000 000`: `700`

## Adjust permission:

- ▶ Users can change the permissions:
  - `chmod 754 filename`
  - `chmod u+wx,g+rx,g-w,o+r,o-wx filename`
- ▶ root can change the ownerships:
  - `chown user:group filename`

# Controlled Invocation

## Superuser privilege is required to execute certain OS functions

- ▶ Example: password changing
  - User passwords are stored in the file `/etc/shadow`
  - This file is owned by the root superuser. A normal user has no access to it
  - When a normal user wants to change his password with the program `passwd`, this program needs to give him additional permissions to write to `/etc/shadow`

## SUID: a special permission flag for a program

- ▶ If SUID is enabled, then user who executes this program will inherit the permissions of the program's owner.
- ▶ A normal user executing `passwd` can get additional root permission to write the new password to `/etc/shadow`

The execute permission of the owner is given as **s** instead of **x**

```
root@cx4062:~# ls -al /usr/bin/passwd
-rwsr-xr-x 1 root root 59976 Mar 14 08:59 /usr/bin/passwd
```

# Security of Controlled Invocation

---

## Many other SUID programs with the owner of root

- ▶ `/bin/login`: login; `/bin/at`: batch job submission; `/bin/su`: change UID

## Potential dangers

- ▶ As the user has the program owner's privileges when running a SUID program, the program should only do what the owner intended
- ▶ By tricking a SUID program owned by root to do unintended things, an attacker can act as the root

## Security consideration

- ▶ All user input (including command line arguments and environment variables) must be processed with extreme care
- ▶ Programs should have SUID status only if it is really necessary.
- ▶ The integrity of SUID programs must be monitored.

# Logging, Monitoring & Auditing

---

## Purposes

- ▶ Intrusion detection: identify unauthorized access or system changes.
- ▶ Forensics and investigation: provide historical data for incident response.
- ▶ Accountability: track user actions and commands.
- ▶ Performance monitoring: assist in debugging applications and diagnosing.

## Challenges

- ▶ High storage and processing requirements: precisely select and record the most critical data.
- ▶ Attackers may erase or modify logs: well protect the data, e.g., via encryption and access control.
- ▶ May compromise user privacy: follow the compliance and retention policies.



# Examples of Monitored Data

---

The OS collects different types of data at different layers.

- ▶ System call traces: describe the activities or behaviors of processes running in the system.
- ▶ Log file: information on user activity, including user' login record, history of commands, etc.
- ▶ File integrity checksums: periodically scan critical files for changes and compare cryptographic checksums for these files, with a record of known good values.
- ▶ Registry access: monitor access to the registry. This is specific to Windows operating systems.
- ▶ Kernel and driver-level monitoring: this source provides insight into OS kernel-level anomalies.
- ▶ Resource usage: CPU, memory or I/O utilization and activities can indicate the execution of some malicious behaviors.
- ▶ Network activities: include established connections and received packets

# Intrusion Detection

---

## Intrusion Detection System (IDS)

- ▶ A system used to detect unauthorized intrusions into computer systems.
- ▶ IDS can be implemented at different layers, including network-based IDS, host-based IDS.
- ▶ We mainly focus on host-based IDS, which monitors the characteristics of a single host for suspicious activities.

## An IDS comprises three logical components:

- ▶ Sensors: responsible for collecting data.
- ▶ Analyzers: responsible for determining if an intrusion has occurred, and the possible evidence. It may provide guidance about what actions to take as a result of the intrusion.
- ▶ User interface: enables a user to view output from the system or control the behavior of the system.

# Detection Methodologies

---

## Signature-based detection

- ▶ Flag any activity that matches the structure of a known attack
- ▶ It is *blacklisting*: keep a list of patterns that are not allowed, and alert if we see something on the list.
- ▶ Advantage: simple and easy to build; good at detecting known attacks.
- ▶ Disadvantage: cannot catch new attacks without a known signature.

## Anomaly-based detection

- ▶ Develop a model of what normal activities look like. Alert on any activities that deviates from normal activities.
- ▶ It is *whitelisting*: keep a list of allowed patterns, and alert if we see something that is not on the list.
- ▶ Advantage: can detect attacks we have not seen before.
- ▶ Disadvantage: false positive rate can be high (many non-attacks look unusual).

# Outline

---

- ▶ **Security Protection Stages in OS**

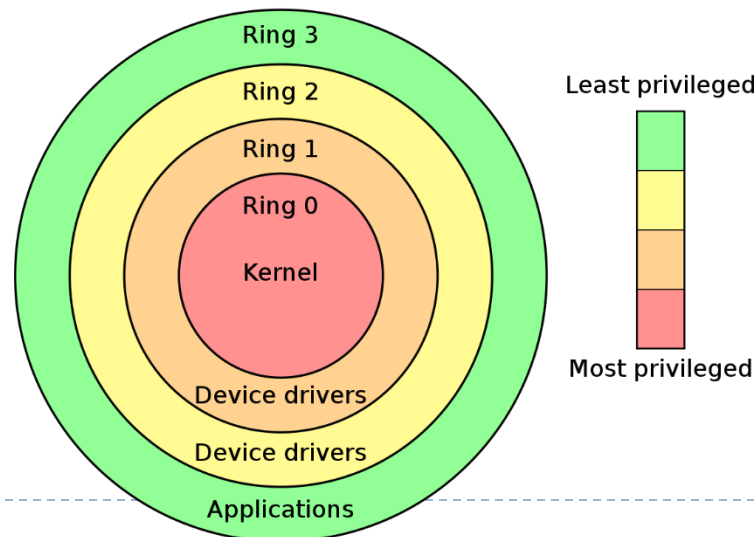
- ▶ Authentication
- ▶ Authorization with Access Control
- ▶ Logging, Monitoring & Auditing

- ▶ **Privilege Management in OS**

# Privileged Rings Inside OS

## Operating modes

- ▶ Kernel mode has the highest privilege, running the critical functions and services; user mode has the least privilege.
- ▶ Entities in the higher rings cannot call the functions and access the objects in the lower rings directly.
- ▶ Context switch is required to achieve the above procedure, system call, interrupt, etc.
- ▶ Status flag allows system to work in different modes.



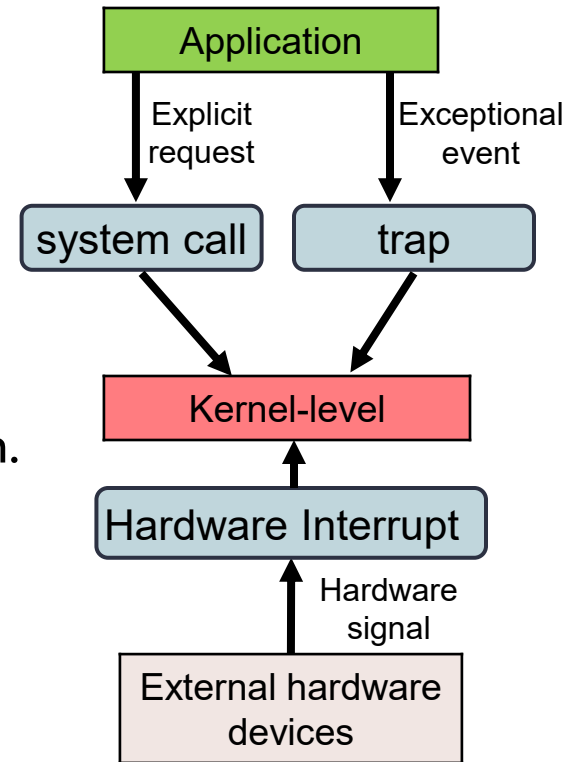
# Context Switch

## Different events can trigger the transition from user to kernel levels

- ▶ System call: user application explicitly makes a request to kernel for privileged operations
- ▶ Trap: user application gets an exceptional event or error and requests the kernel to handle.
- ▶ System call and trap belong to software interrupts,
- ▶ Hardware interrupt: hardware issues a signal to the CPU to indicate an event needs immediate attention.

## Switch procedure

- ▶ CPU stores process's states, and switches to the kernel mode by setting the status flag.
- ▶ Kernel handles the interrupt based on the interrupt vector in an interrupt table.
- ▶ CPU switches back to user mode and restores states



# How System Call is Issued and Handled

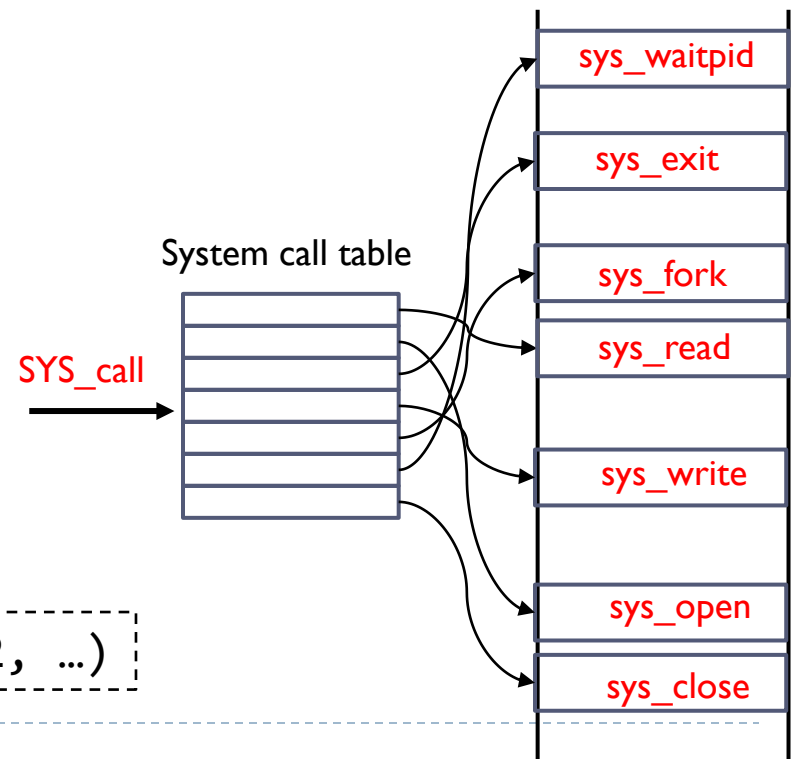
A system call is an interface that allows a user-level process to request functions or services from the kernel level.

- ▶ Process control
- ▶ File management
- ▶ Device management

## How to issue a system call?

- ▶ System call table: a table of pointers in the kernel region, to different system call functions.
- ▶ A user process passes the index of the system call and parameters with the following API:

```
syscall(SYS_call, arg1, arg2, ...)
```



# Rootkit

---

## Malware that obtains root privileges to compromise the computer

- ▶ Root user does not go through any security checks, and can perform any actions to the system
  - Insert and execute arbitrary malicious code in the system's code path
  - Hide its existence, e.g., malicious process, files, network sockets, from being detected.

## How can the attacker gain the root privileges?

- ▶ Vulnerabilities in the software stack: buffer overflow, format string...

There are some common techniques for rootkits to compromise the systems.



# Highjack System-call Table

Rootkit changes pointers of certain entries in the system-call table.

- ▶ Other processes calling these system calls will execute the attacker's code

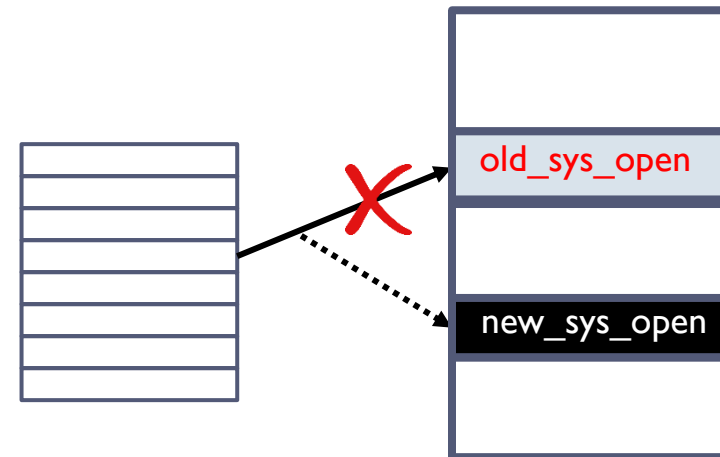
## Example

- ▶ `syscall_open` is used to display the running process (`ps` command)
- ▶ Rootkit redirects this system call to `new_syscall_open`
  - When the object to be opened matches the malicious name, return NULL to hide it
  - Otherwise, call normal `old_syscall_open`

```
1 struct file sysmap = open("System.map-version");
2 long *syscall_addr = read_syscall_table(sysmap);

4 old_syscall_open = syscall_addr[__NR_open];
5 syscall_addr[__NR_open] = new_syscall_open();

7 malicious_object_name = {"xingyi", "bind_shell",
8                           "reverse_shell"...};
8 int new_syscall_open(char *object_name) {
9     if strstr(object_name, malicious_object_name)
10         return NULL;
11     return old_syscall_open(object_name)
12 }
```



# Compromise System Call Functions

Rootkit can also directly change the system call function.

## Example

- ▶ Replace the first 7 bytes of `syscall_open` as jump to `malicious_open`.
  - This faked system call will issue malicious function, restore the original system call and then call the correct one.

```
1 struct file sysmap = open("System.map-version");
2 long *syscall_addr = read_syscall_table(sysmap);
3 syscall_open = syscall_addr[__NR_open];

5 char old_syscall_code[7];
6 memcpy(old_syscall_code, syscall_open, 7);

8 char pt[4];
9 memcpy(pt, (long)malicious_open, 4);
10 char new_syscall_code[7] =
11 {"\xbd", pt[0], pt[1], pt[2], pt[3], // movl %pt, %ebp
12 "\xff", "\xe5"};                // jmp %ebp
13 memcpy(syscall_open, new_syscall_code, 7);

15 int malicious_open(char *object_name) {
16     malicious_function();
17     memcpy(syscall_open, old_syscall_code, 7);
18     return syscall_open(object_name);
19 }
```

