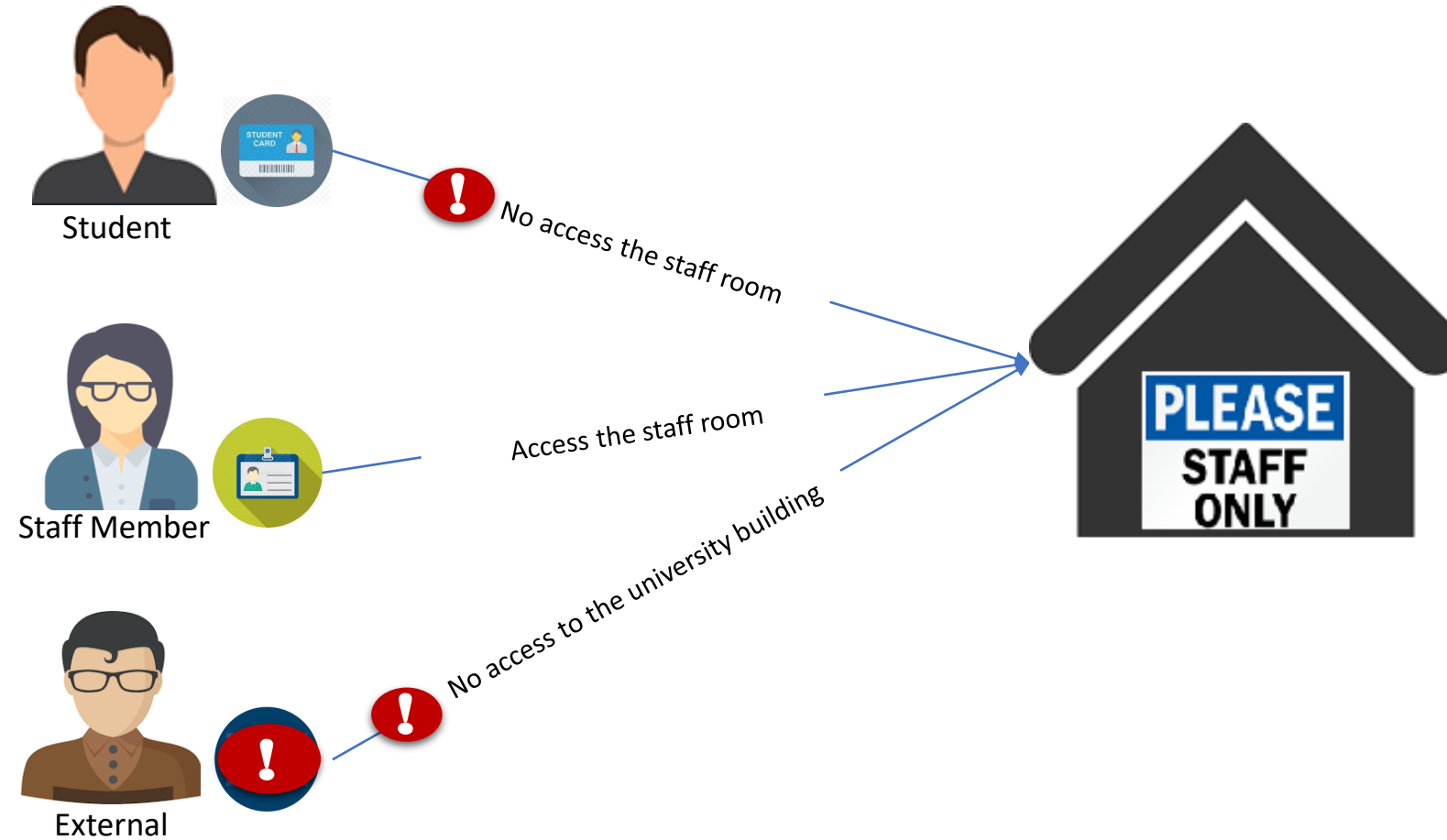


Access Control

Dr. Sana Belguith

Access Control Example



Access Control Example

Identification



Student



Staff Member



External

Present Identity

Authentication



Student



Staff Member



External

Present Photo ID

Enforcement Mechanism



Check ID by Security Agent

Authorisation



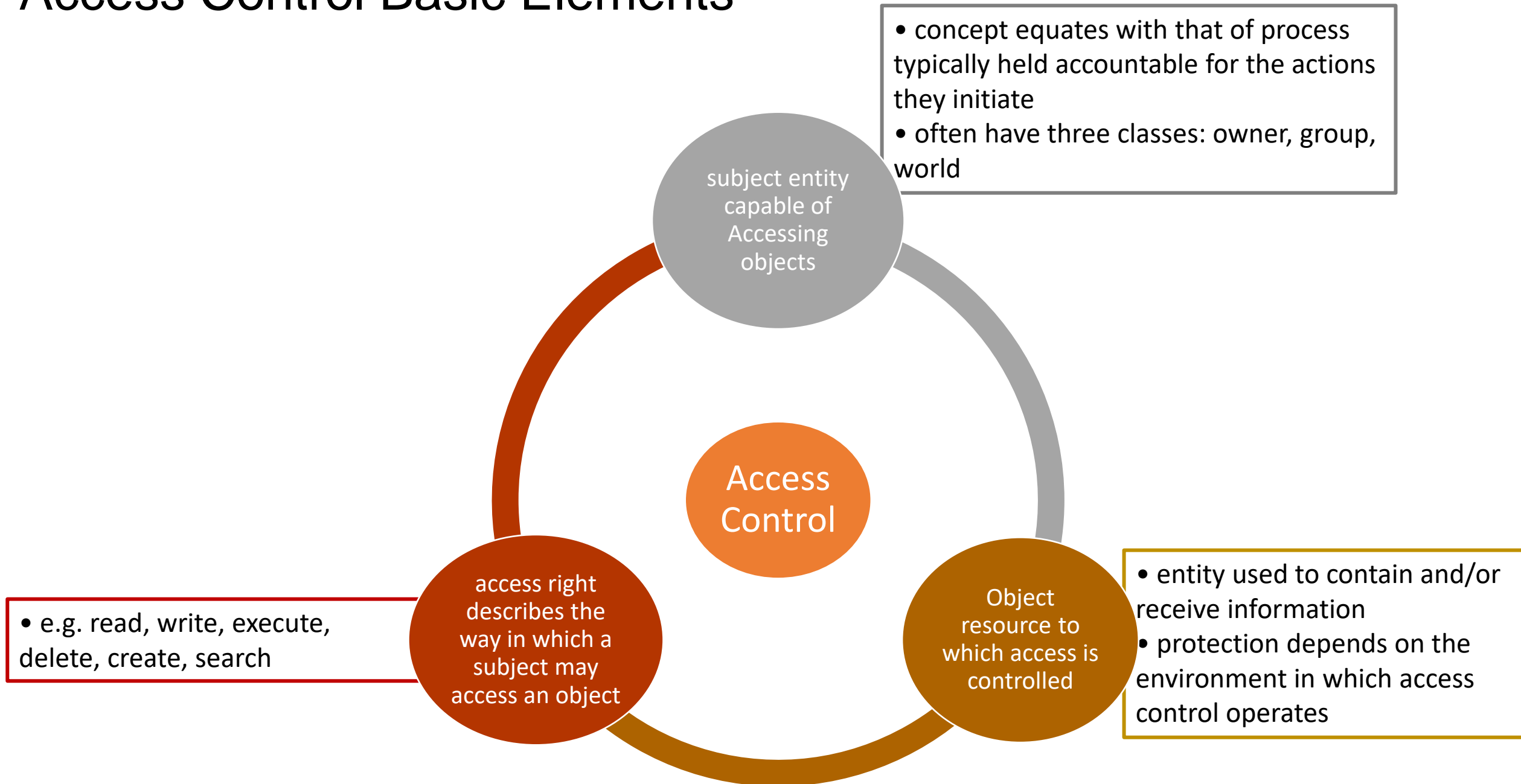
Staff Member Access

What is Access Control



- The ability to allow only authorized users, programs or processes system or resource access
- The granting or denying, according to a particular security model, of certain permissions to access a resource
- An entire set of procedures performed by hardware, software and administrators, to monitor access, identify users requesting access, record access attempts, and grant or deny access based on pre-established rules.

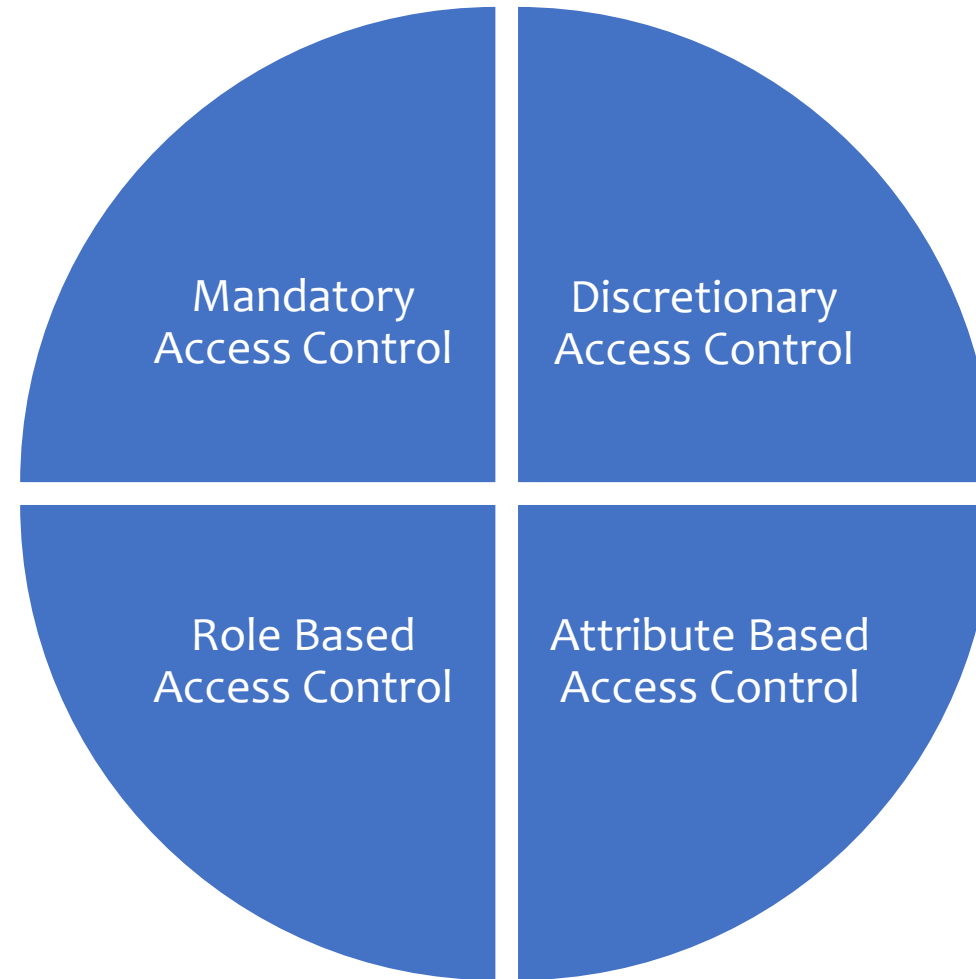
Access Control Basic Elements



Access Control: Domains and Applications

- Social Networks
- Web Browsers
- Operating Systems
- Firewalls

Access Control Methods



Multiple Access Control Policies. DAC, MAC, RBAC are not mutually exclusive.
A system may implement two or even three of these policies for some or all types of access.

Discretionary Access Control (DAC)

- scheme in which an entity may enable another entity to access some resource
- often provided using an access matrix
 - one dimension consists of identified subjects that may attempt data 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
- rule enforcement may be waived or modified by some users.

Access Control Matrix

- ACM is a matrix of all principals and objects.
- The matrix entries describe the permissions for Subjects, Objects, and operations
- Can determine
 - Who can access an object
 - What objects can be accessed by a subject
 - What operations a subject can perform on an object
- Problem: maintaining such a matrix can be difficult.
- If the matrix is corrupted then all controls is lost.

Access Control Matrix

- Suppose the private key file for J is object O1: Only J can read O1
- Suppose the public key file for J is object O2: All can read, only J can modify
- Suppose all can read and write from object O3

- What's the access matrix?

	O1	O2	O3
J	R	R, W	R, W
S2	N	R	R, W
S3	N	R	R, W

Least Privilege

- The Principle of Least Privilege states that a subject should be given only those privileges needed for it to complete its task.
- If a subject does not need an access right, the subject should not have that right.

	O1	O2	O3
J	R	R, W	N
S2	N	R	R, W
S3	N	R	R, W

- Limit permissions to those required and no more
- Consider three processes for user J
- Restrict privilege of the process O3 to prevent leaks

Access Control Lists (ACLs)

- We do not want to store one massive matrix.
- Instead we can store each column of the matrix with the object it refers to.
- For example:
(O2,(J,RW), (S2,R), (S3,R))

	O1	O2	O3
J	R	R, W	R, W
S2	N	R	R, W
S3	N	R	R, W

Mandatory Access Control (MAC)

- Systems administrator sets the policy as to who can access what
- Typical implementations of you might see:
 - SELinux: NSA's patches for Linux to add a MAC. Complex but powerful.
 - AppArmor: Canonical's simplified MAC system based on paths
 - TOMOYO: another simplified system based on paths
- Policies can be applied to more than just files

For example: “*The web browser can access files in the downloads folder ONLY*”
- Rules are enforced on every attempted access, not at the discretion of any system user;

Role-Based Access Control

- Role-based access control (RBAC) is a widely used security framework claimed to be especially appropriate for commercial settings.
- Unlike access control policies that assign permissions to subjects, RBAC associates permissions with functions/jobs/roles within an organization.
- A role is a collection of job functions. Roles within a bank might include: president, manager, trainer, teller, auditor, janitor, etc.

Role-Based Access Control

Associate permissions with job functions

- Each job defines a set of tasks
- The tasks need permissions
- The permissions define a role
- Bank Teller:
 - Read/Write to client accounts
 - Cannot create new accounts
 - Cannot create a loan
- Role defines only the permissions allowed for the job

Role-Based Access Control

An individual has:

- a set of authorized roles, which it is allowed to fill at various times;
- a set of active roles, which it currently occupies.

Roles have an associated set of transactions, which are the activities that someone in that role is permitted to carry out.

- The set of transactions can be organization specific: open an account, cash a check, transfer funds, etc.

Role-Based Access Control

- The following are the three primary RBAC rules:
 - Role assignment: A subject can execute a transaction only if the subject has an active role.
 - Role authorization: A subject's active role must be an authorized role for that subject.
 - Transaction authorization: A subject can execute a transaction only if the transaction is authorized for one of the subject's active roles.
- Note that a subject can have multiple roles. For example, a bank president might also act as a teller.

Attribute-based Access Control

- In the attribute-based access control mechanism, access rights are defined upon attributes.
- The users' identities are not used anymore to define access rights.
- In ABAC, to get access, users need to prove their possession of the required attributes. Thus, users are authenticated to the server to prove their rights.
- ABAC is an extension of RBAC where users' roles are more generalized.
- User access rights are expressed using a set of attributes.

Traditional UNIX File Access Control

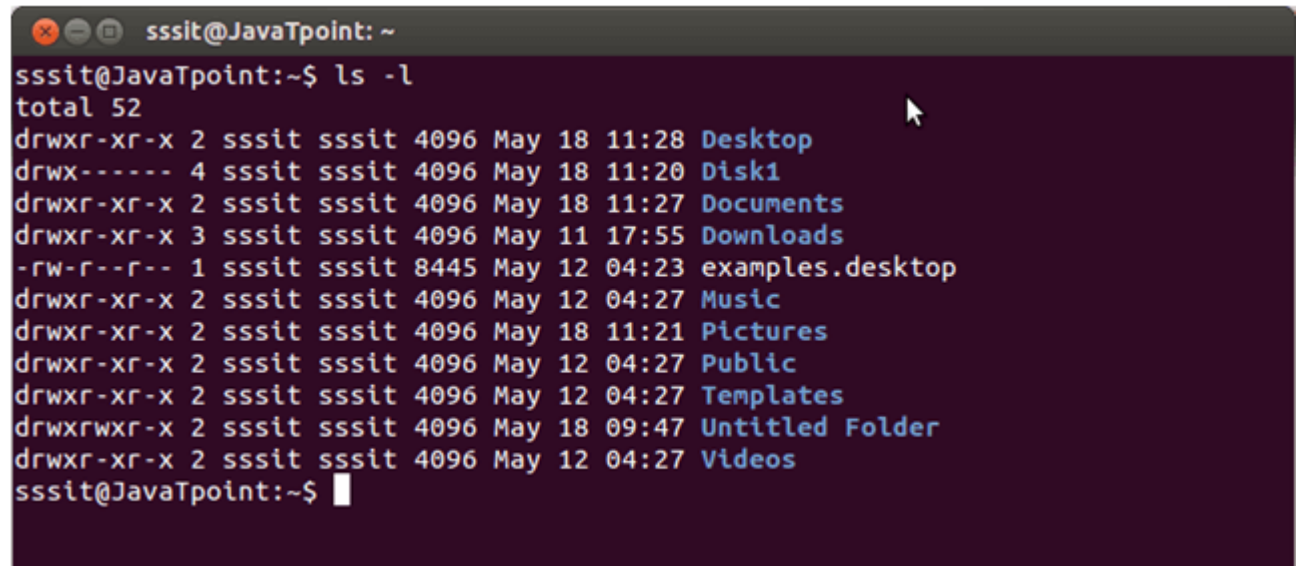
- “set user ID”(SetUID)
- “set group ID”(SetGID)
 - system temporarily uses rights of the file owner / group in addition to the real user’s rights when making access control decisions
 - enables privileged programs to access files / resources not generally accessible
- superuser
 - is exempt from usual access control restrictions
 - has system-wide access

Access Control Lists (ACLs) in UNIX

- modern UNIX systems support ACLs: FreeBSD, OpenBSD, Linux, Solaris
- FreeBSD
 - Setfacl command assigns a list of UNIX user IDs and groups
 - any number of users and groups can be associated with a file
 - read, write, execute protection bits
 - a file does not need to have an ACL
 - includes an additional protection bit that indicates whether the file has an extended ACL
- when a process requests access to a file system object two
- steps are performed:
 - step 1 selects the most appropriate ACL: owner, named users, owning / named groups, others
 - step 2 checks if the matching entry contains sufficient permissions

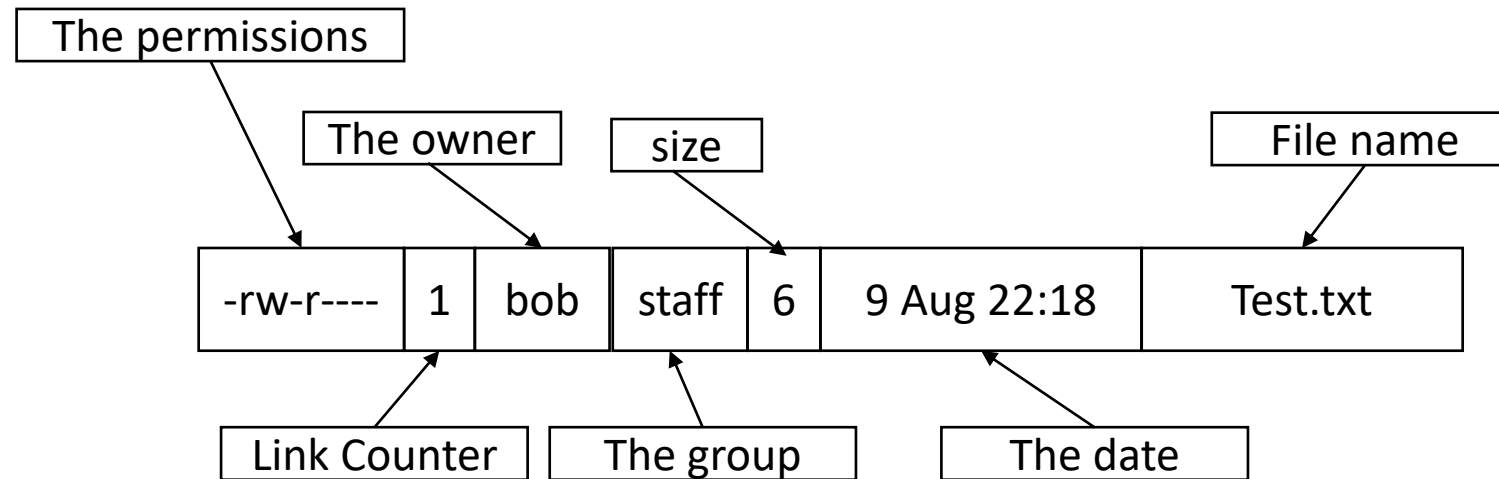
Access Control in Unix/Linux

- Unix/Linux/Mac use ACL, with groups.
- “uid” set when you log on.
- Linux Kernel then dynamically enforces the ACLS.
- `ls -l` displays files with their ACL
- `root` owns everything (“get root” = control the system)



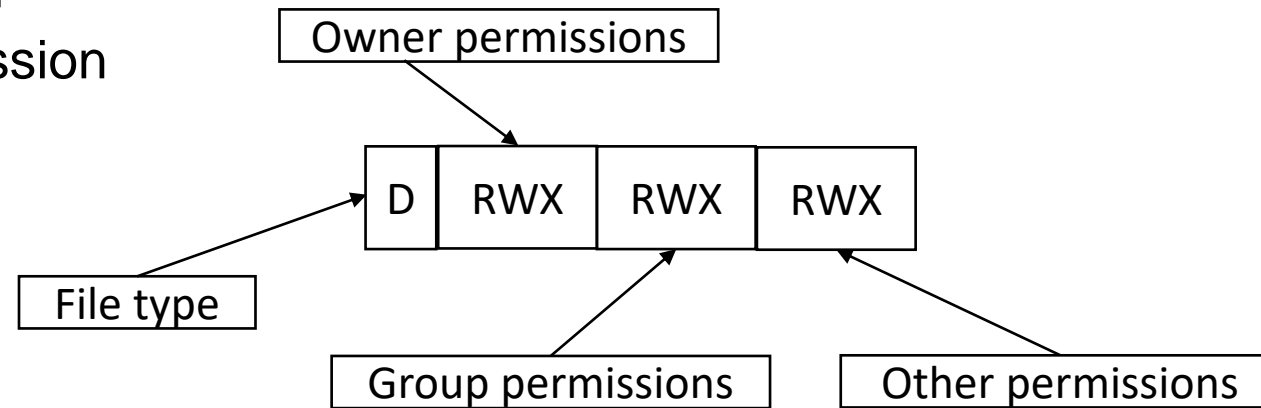
```
sssit@JavaTpoint: ~  
sssit@JavaTpoint:~$ ls -l  
total 52  
drwxr-xr-x 2 sssit sssit 4096 May 18 11:28 Desktop  
drwx----- 4 sssit sssit 4096 May 18 11:20 Disk1  
drwxr-xr-x 2 sssit sssit 4096 May 18 11:27 Documents  
drwxr-xr-x 3 sssit sssit 4096 May 11 17:55 Downloads  
-rw-r--r-- 1 sssit sssit 8445 May 12 04:23 examples.desktop  
drwxr-xr-x 2 sssit sssit 4096 May 12 04:27 Music  
drwxr-xr-x 2 sssit sssit 4096 May 18 11:21 Pictures  
drwxr-xr-x 2 sssit sssit 4096 May 12 04:27 Public  
drwxr-xr-x 2 sssit sssit 4096 May 12 04:27 Templates  
drwxrwxr-x 2 sssit sssit 4096 May 18 09:47 Untitled Folder  
drwxr-xr-x 2 sssit sssit 4096 May 12 04:27 Videos  
sssit@JavaTpoint:~$
```

The UNIX Access Control List



UNIX File Permissions

- Permissions:
 - r: read permission
 - w: write permission
 - x: execution permission
 - : no permissions
- File Type:
 - : file
 - d : directory
 - b/c: device file



Sandbox

Sandboxing is a cybersecurity procedure in which you run code, analyze it, and code in a secure, enclosed environment on a system that resembles end-user working environments. It is intended to prevent the potential threat from entering the network and is commonly used to scrutinize unknown or non-secure code.

Running Untrusted Code

We often need to run buggy/untrusted code:

- programs from untrusted Internet sites: toolbars, viewers, codecs for media player
- old or insecure applications
- legacy software

=> Goal: if application “misbehaves,” kill it.

Examples

Web Pages: Your browser essentially sandboxes the web pages it loads. Web pages can run JavaScript code, but this code can not do anything it wants — if JavaScript code tries to access a local file on your computer, the request will fail.

PDFs: Adobe Reader now runs PDF files in a sandbox, preventing them from escaping the PDF viewer and tampering with the rest of your computer.

Examples

Mobile Apps: Mobile platforms run their apps in a sandbox. Apps for iOS, Android, and Windows 8 are restricted from doing many of the things standard desktop applications can do. The sandbox disconnects the applications, stopping them from interfering with one another.

Windows Programs: User Account Control functions as a bit of a sandbox, essentially restricting Windows desktop applications from modifying system files without first asking you permission. User Account Control just restricts access to system files and system- wide settings.

Benefits of Sandboxing

- **Prevents zero-day attacks:**

Zero-day threats exploit unknown security flaws. They are highly hazardous because manufacturers cannot issue patches unless they fully comprehend the exploited vulnerability.

Sandboxing software performs admirably in isolating these menaces. Though there is no assurance that it will prevent zero-day manipulations, sandboxing most often works for damage limitation by distancing the potential danger from the remainder of the network.

- **Enhances other security programs**

Sandboxing is an excellent complement to specific other security software, ranging from action monitoring programs to antivirus programs. Conversely, sandboxing software shields against spyware strains that your virus protection may struggle to detect.

Benefits of Sandboxing

- **Helps examine presumably malicious programs for threats**

If you're dealing with new vendors or untrustworthy software sources, you can try out a new application for threats before incorporating it.

- **Allows rigorous software tests before they are released**

Software changes should be thoroughly tested before they are released to the public. Sandboxing can be used to test new code for possible risks before it officially launches. Therefore, it has a vital role to play in application security.

How To Sandbox Any Program

- **Virtual Machines:** A virtual machine program like VirtualBox or VMware creates virtual hardware devices that it uses to run an operating system.

This entire operating system is essentially sandboxed, as it does not have access to anything outside of the virtual machine.

- **Using other tools:** Sandboxie, Bufferzone

Types of Application Sandboxes

Type A: OS enhancement based: Sandboxie, Buffer Zone Pro etc.

- Custom kernel driver modifies Windows behavior, so that change to protected system components is prevented
- Use cases: Most of such sandboxes are used for controlled execution of applications

Type B: Master/slave model: Adobe ReaderX, Chrome browser

- Slave is confined using OS access control facilities
- Master mediates access to resources
- Use case: protect the application from exploitation

Reading references

- [1] Stallings, W., Cryptography and Network Security: Principles and Practice, 7th ed., 2017, Pearson.
- [2] Goodrich M. and Tamassia Roberto, Introduction to computer security, Boston, MA : Pearson, 2011.
- [3]- Bishop M., Introduction to computer security, Chapter 12, *first printing*, 2004.
- [4] Prevelakis, V. and Spinellis, D., 2001, June. Sandboxing Applications. In USENIX Annual Technical Conference, FREENIX Track (pp. 119-126).
- [5] Greamo, C. and Ghosh, A., 2011. Sandboxing and virtualization: Modern tools for combating malware. IEEE Security & Privacy, 9(2), pp.79-82.