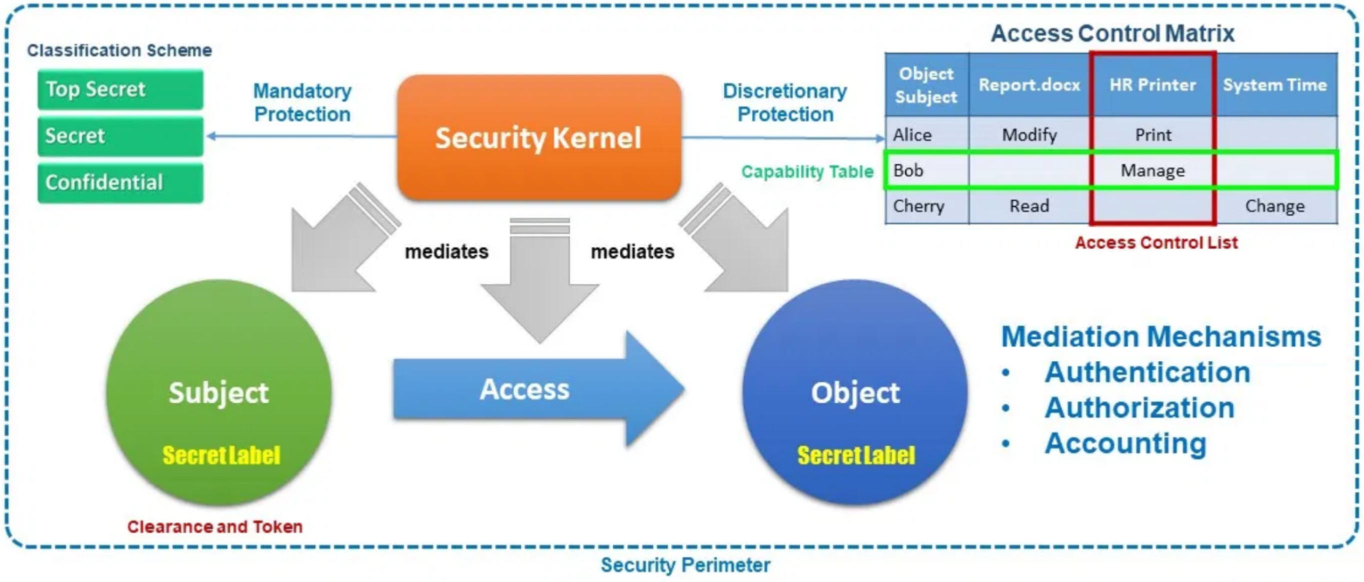
LS-07

OS Protection Concepts. Access Control Models.

Agenda

1. Access Control basic concepts.
2. Access Matrix Implementation.
    - 2.1. Full Access Matrix.
    - 2.2. ACL - Access Control Lists.
    - 2.3. Capability Lists.
    - 2.4. Authorization Relationships.
    - 2.5. Attribute schemes.
3. Access Control Models.
    - 3.1. DAC.
    - 3.2. MAC.
    - 3.3. RBAC.
    - 3.4. ABAC.
    - 3.5. Other Access Control Models.
    - 3.6. CISSP Practice Questions.



Object Subject	Report.docx	HR Printer	System Time
Alice	Modify	Print	
Bob		Manage	
Cherry	Read		Change

Mediation Mechanisms
- Authentication
- Authorization
- Accounting

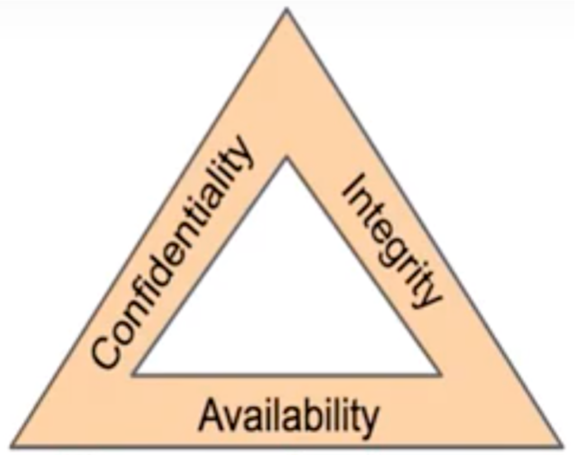
1. Access Control basic concepts.

1.1. Basic Security Theorem – any activity will always result in a secure state.

1.2. Guiding principle - principle of least privilege.

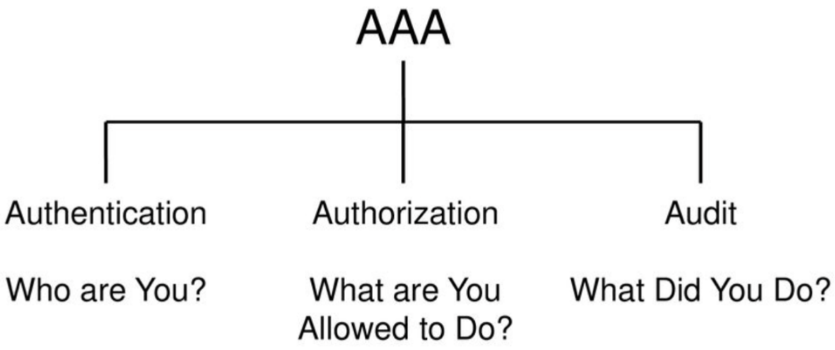
1.3. CIA-triangle - three main requirements of Information Security are:

- Confidentiality - only authorized persons should access the data.
- Integrity - only authorized persons can change the data.
- Availability - information is available to authorized persons when they need.

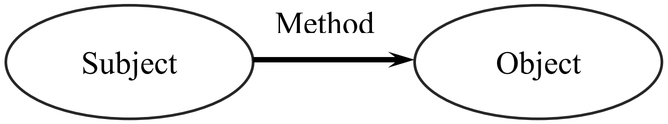


1.4. 3Au Security Principe's.

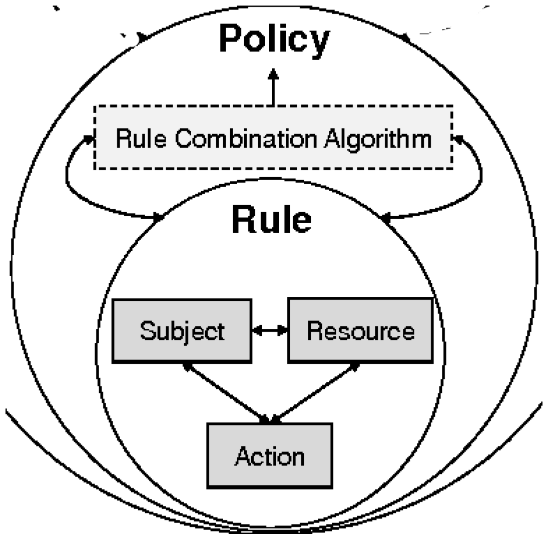
- Authentication,
- Authorization,
- Audit (Acounting).



1.5. Basic Security Process (Rule) – Subjects can access Objects (Resources) using access Rights (Actions, Methods).



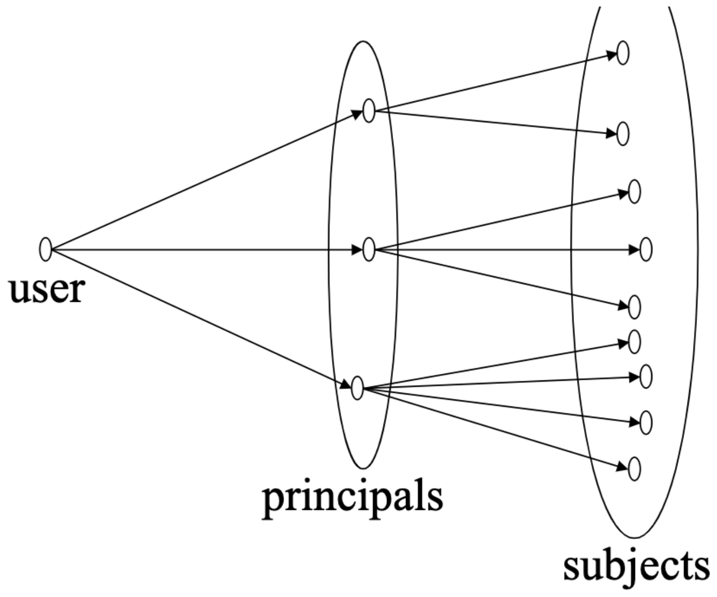
1.6. Policy – Rule Combination.

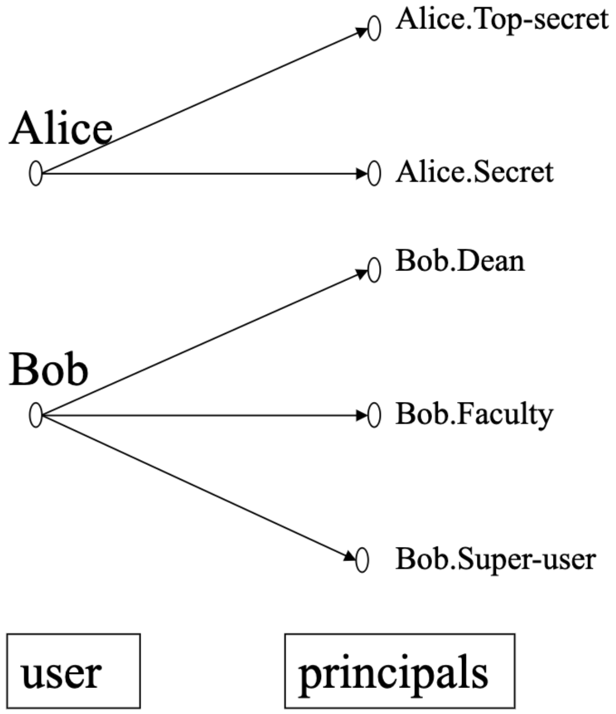


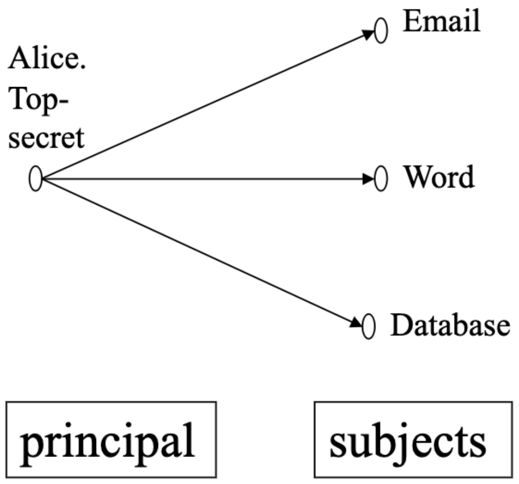
1.7. Subject is any entity capable of initiating operations on objects: a process of physical, logical, system users. Subjects not are users.

User – Principal – Subject.

- A human user may manifest as multiple users (accounts, principals) in the system
- System authenticates user in the context of principal
- Shared principals (accounts) are not good for accountability
- A subject is a program (application) executing on behalf of a principal
- Subjects are often treated the same as principal if all subjects of a principal have the same rights







1.8. Object is a part of an information system, access to which is limited by a security policy. Each object has a unique name that distinguishes it from other objects in the system.

Object types:

1. Objects corresponding to hardware resources: processor, memory, peripheral device, disk, network.
2. Objects corresponding to information resources: document, mail, graphic, video, sound.
3. Logical objects of system software (OS resources): thread, process, memory page, socket, port, file, named and anonymous pipes, message queues, semaphores, and shared memory pages, user list, set of security policies and models, settings registry equipment.
4. Logical objects of the application software: admin settings, user settings.
5. Database objects: table, column, row or field value.

1.9. Access Method is a type of actions performed on an object. Examples by Objects in the table.

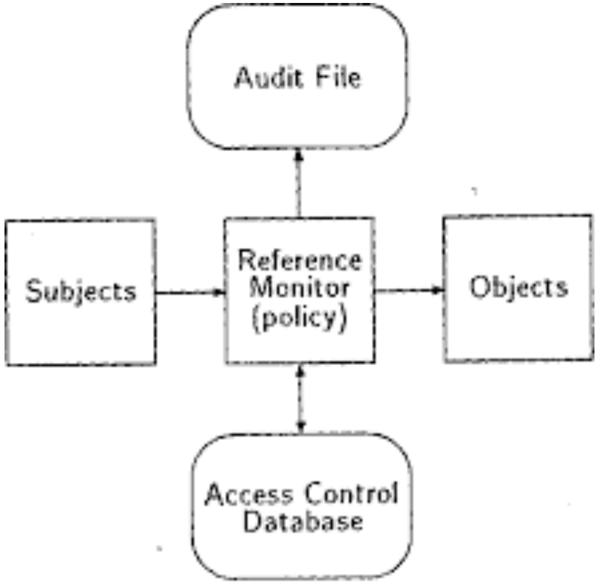
Access Object	**Access Methods (Operations)**
Memory Page	Reading, writing data; execution of program code; division of memory, including mapping to the memory of system devices and copy-by-write.
Process	Process creation; completion of the process; suspension of a process; kill.
File	Reading, rewriting, executing a file; reading information about a file; appending information to a file; opening a file; dumping buffers, moving the position pointer; getting the current value of the position pointer.
Database	Database connection; adding a table; deleting a table; operations with database tables and table records; viewing the transaction log.
All	Owner, create, delete, transfer, grant

1.10. Object Owner - the entity who is responsible for the Confidentiality, Integrity and Availability of the object. The owner grants access to the object to any other subject. Typically, the Subject who created a given Object is automatically assigned its Owner. In the future, the owner can be changed using the appropriate accessor Method to the object.

1.11. Access Control Systems implement the security process and are based on the concept of an Access Monitor.

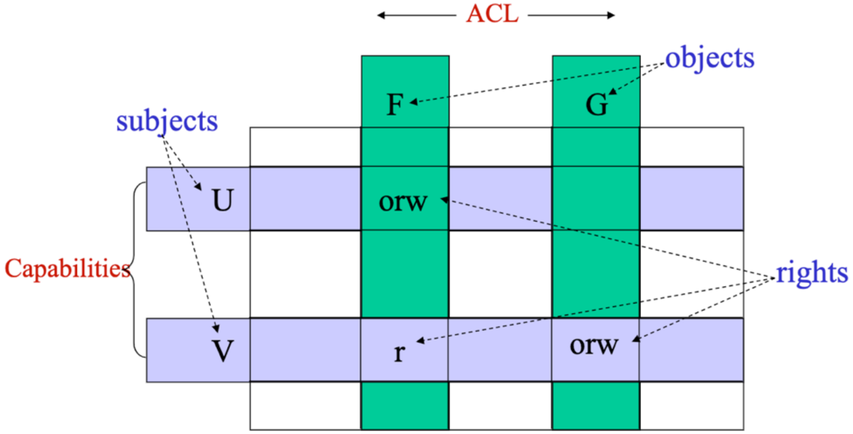
Access Monitor is a module that mediates all requests of Subjects to Objects. The Access Monitor uses a Access Control Database that stores access control rules and, based on this information, allows or does not allow the Subject to access the Object, and also records information about the access attempt in the system log (Audit File).

Access Control Database can be built on the basis of an Access Matrix or one of its representations.



1.12. Access Control Model is a specification of a security policy, which is described by the rules of the subject's access to information system objects.

1.13. Access Matrix – is a table in which rows correspond to subjects, columns correspond to objects, and the intersection of a row and a column contains Rights for the subject to access the object.

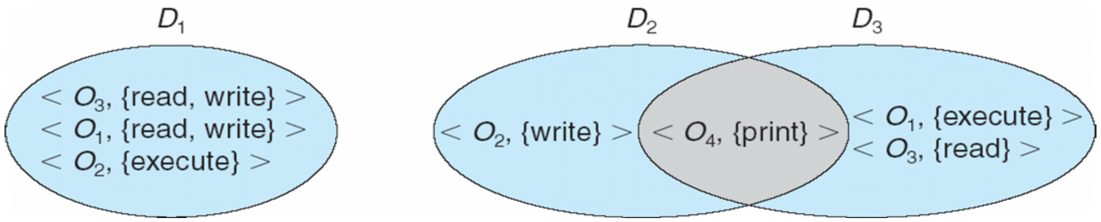


2. Access Matrix Implementation.

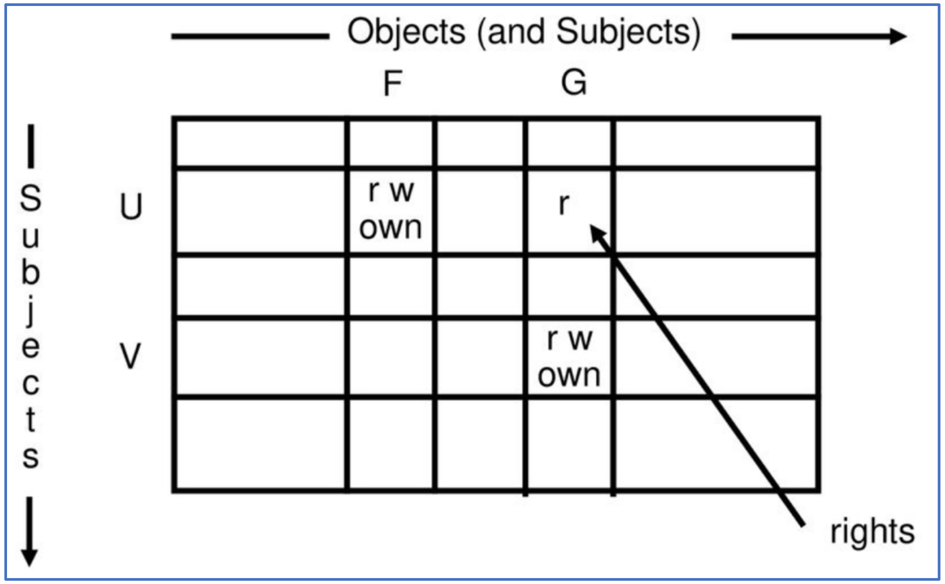
2.1. Full Access Matrix.

Access Rights = Object Name + Rights Set, where *rights-set* is a subset of all valid operations that can be performed on the object.

Domain = Set of Access Rights



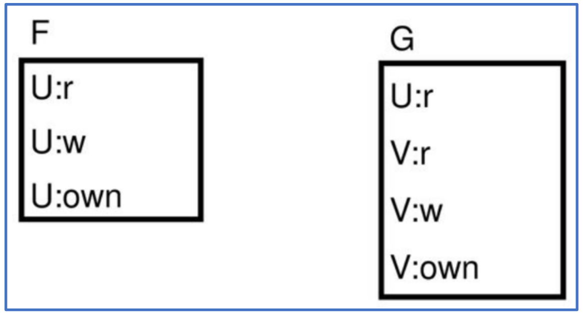
Access Matrix – is a table in which rows correspond to Subjects (Domains), columns correspond to Objects or other subjects, and the intersection of a row and a column contains Rights (rules of permissions) for the subject to access the object (or other subject).



The main disadvantages of this matrix is its excessively large dimension and complexity of administration; all interconnections and limitations of the domain have to be taken into account manually.

To overcome these difficulties the access matrix is often replaced by some of its implicit ideas.

2.2. ACL - Access Control Lists.



F

U:r
U:w
U:own

G

U:r
V:r
V:w
V:own

Use on Linux, Windows. Each column of the access matrix is stored with the object corresponding to that column.

ACL requires Subjects to be authenticated before access to a particular object.

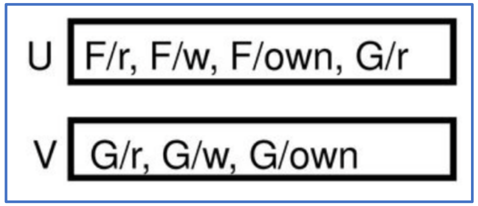
Advantages ACL:

For each object, a list of subjects that have non-zero access rights to them (indicating these powers) is specified. As a result, memory is seriously saved, since all zero values that make up most of it are excluded from the access matrix.

Disadvantages of ACL:

- ACL takes up a varying amount of space.
- The problem of inheritance of permissions for the subjects.
- The problem of finding objects allowed by the subject.
- When deleting a subject, the object may be accessible to a non-existent subject.

2.3. Capability Lists (Списки полномочий субъектов).



U F/r, F/w, F/own, G/r

V G/r, G/w, G/own

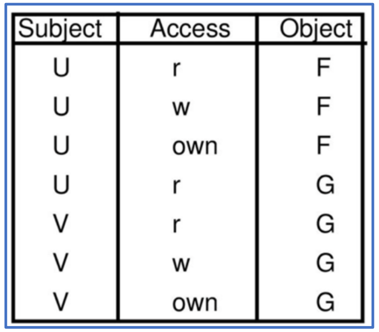
Use on Kerberos. It is similar to ACL with the only difference that for each subject there is a list of objects, access to which is allowed with the indication of access permissions.

This view is called the subject profile.

Capabilities do not require Subjects to be authenticated.

2.4. Authorization Relations.

Access matrix can also be implemented/stored in relation form. This form is often used in database management systems.



Subject	Access	Object
U	r	F
U	w	F
U	own	F
U	r	G
V	r	G
V	w	G
V	own	G

2.5. Lock-Key Pare (Attribute) scheme.

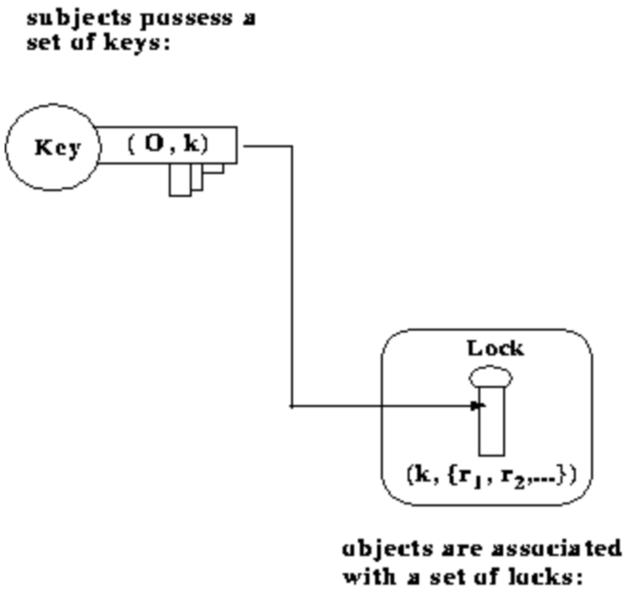
LKP scheme proposed by Chang and Jan (Thaiwan, 1991) for frequently inserted or deleted users and files.

Can be used for CISCO Firewall Dynamic Access Lists or for Crypto Currency.

- Each object has list of unique bit patterns, called locks.
- Each subject as list of unique bit patterns called keys.
- Keys can be passed freely from subject to subject, easy revocation.

Process in a subject can only access object, if subject has key that matches one of the locks.

The elements of the access matrix are not stored explicitly, but are dynamically calculated on each access attempt for a particular subject-object pair based on their attributes.


subjects possess a set of keys:
Key
(O , k)
Lock
(k, {r1, r2,...})
objects are associated with a set of locks:

Advantages LKP.

- Saving memory.
- Consistency of the protection database is achieved.
- Ease of administration.

Disadvantages LKP. The main disadvantage is the complexity of setting the access rights of a specific subject to a specific object.

Example. ACLs implementation in POSIX systems

Problem: an ACL takes up a varying amount of space (possibly a lot!), won’t fit in an inode.

UNIX Compromise:

1. A file defines access rights for three domains: the owner, the group and everyone else
2. Extended ACLs: stored outside of the inode Enumerated list of permissions on users and groups
3. Permissions (Rights): Read, write, execute (directory search)
4. Default permissions set by the umask system call
5. chown system call changes the object’s owner
6. chmod system call changes the object’s permissions
7. Extended attributes stored outside of the inode (noatime, append-only, immunization, nodump, …)
8. Operations on all objects: delete, readattr, writeattr, readextattr, writeextattr, readsecurity, writesecurity, chown
9. Operations on directories: list, search, add_file, add_subdirectory, delete_child
10. Operations on files: read, write, append, execute
11. Escalation (not propagation) of Rights: Set user ID on execution (SUID)
12. Inheritance controls: Set group ID on execution (SGID)

3. Access Control Models.

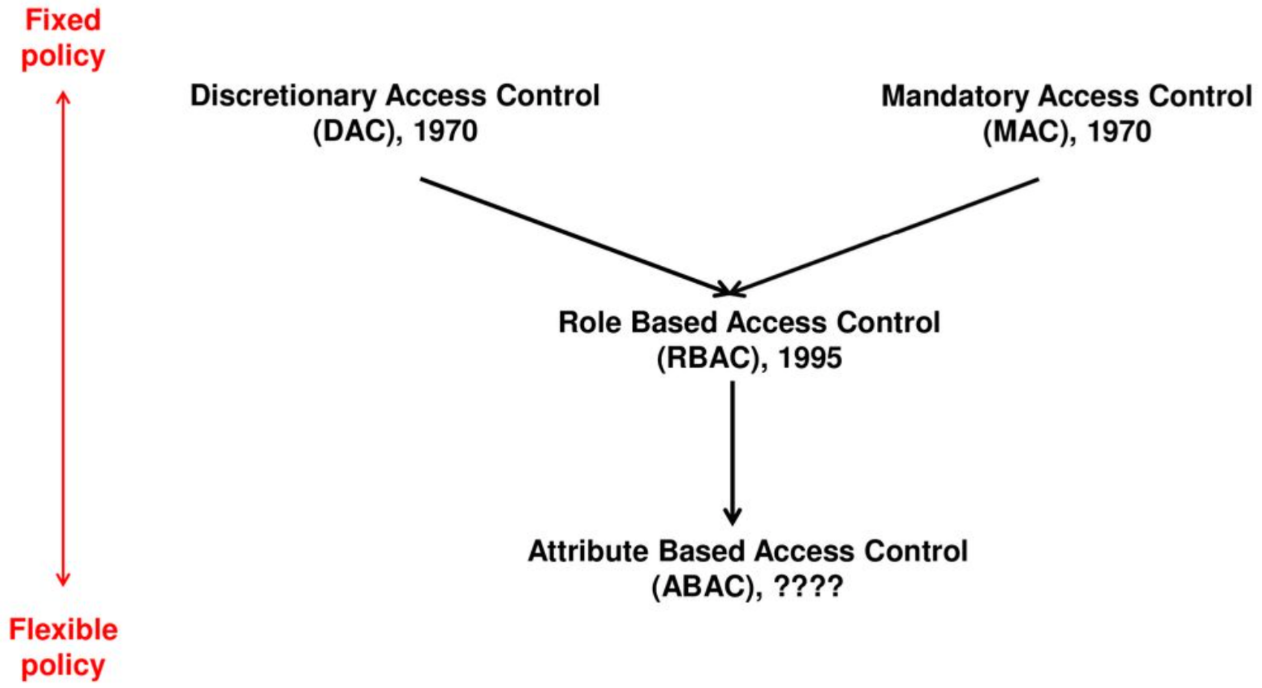
Access Control Model is a specification of a security policy, which is described by the rules of the subject's access to information system objects.

There are many theoretical models of access control:

- Harrison-Ruzzo-Ulman Model
  (full access matrix for objects, subjects and access rights for confidentiality and integrity);
- Bell-LaPadulla Model (BLP)
  (first use in DoD for military information confidentiality);
- Biba Model
  (use for military/government information integrity);
- Clark-Wilson Model
  (commercial integrity – base on rule of least privilege and non-DAC – transaction with second subject);
- Brewer-Nash Model
  (or Chinese Wall, or Conflict of interest prevention);
- Graham-Denning Access Control Model
  (define 8 protection rights: create & delete objects & subjects, read, grant, delete, transfer access);
- Take-Grant Model
  (formal model used to analyze DAC systems)
- Many more.

Have four main access control realization:

1. **DAC** – Discretionary Access Control (based on access matrix, noncentral management support).
    - IBAC – Identity Based Access Control (DAC sub-model).
2. **MAC** –Mandatory Access Control (based on security labels, central management support).
    - LBAC – Lattice (решётка) Based Access Control (MAC sub-model).
3. **RBAC** – Role Based Access Control (in Windows use groups to provide RBAC).
    - TBAC – Task Based Access Control (RBAC sub-model).
4. RuBAC – Rule Based Access Control (based on global rules, filters, deny/allow, ex. Firewall).
    - **ABAC** – Attribute Based Access Control (RuBAC sub model) (a next generation authorization model, based on control subj/obj characteristics: ip-addr., time, location, language, device, etc).

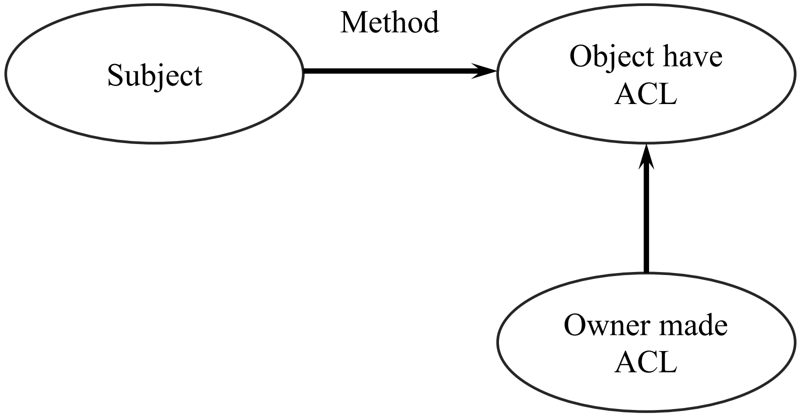

Fixed policy
Discretionary Access Control (DAC), 1970
Mandatory Access Control (MAC), 1970
Role Based Access Control (RBAC), 1995
Attribute Based Access Control (ABAC), ????
Flexible policy

3.1. DAC - Discretionary Access Control.

Since 1970.

DAC based on the Access Control Matrix. Owner set object permissions.

DAC use decentralized access control. Permission rule attached to the Object.



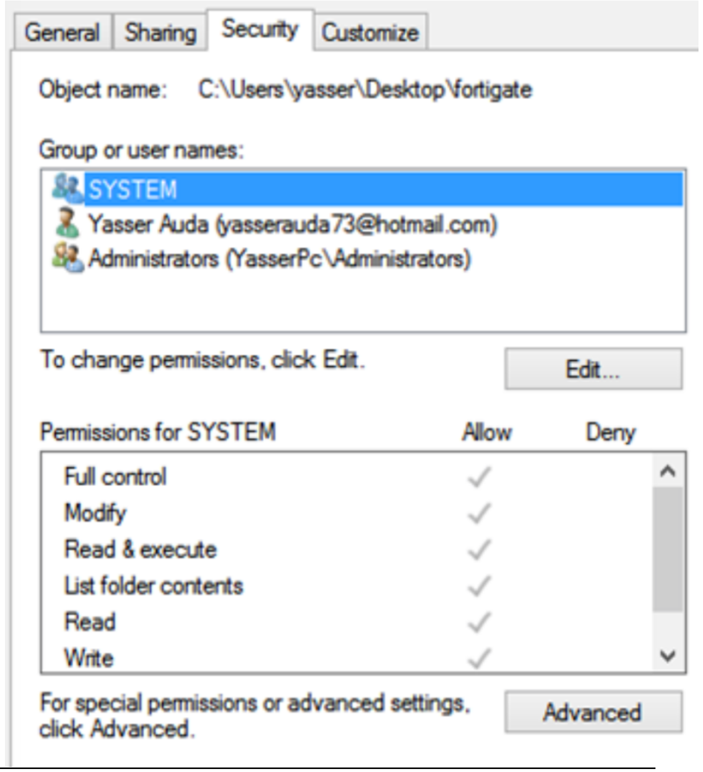
If the subject provides credentials that Access Monitor match what is contained in the ACL, the subject is granted access to the object.

3.1.1. Implementation.

DAC is implemented in common Unix and Windows operating systems, the owner can arbitrarily change the access rights to the object he created, for example, to make it public.

3.1.2. Disadvantages:

- Each object needs its own ACL and a set of privileges assigned to each subject.
- Adding subjects to the ACL is static and assigned by the owner of the object.
- Without the periodic process of removing or revoking access, users accumulate privileges.
- Trojan Horse problem.



Object name: C:\Users\yasser\Desktop\fortigate

Group or user names:

SYSTEM
Yasser Auda (yasserauda73@hotmail.com)
Administrators (YasserPc\Administrators)

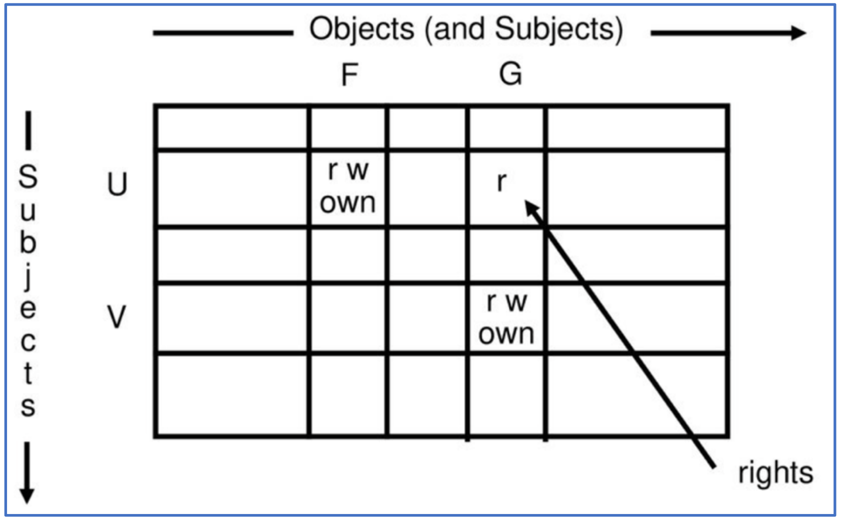
To change permissions, click Edit.

Permissions for SYSTEM	Allow	Deny
Full control	✓	
Modify	✓	
Read & execute	✓	
List folder contents	✓	
Read	✓	
Write	✓	

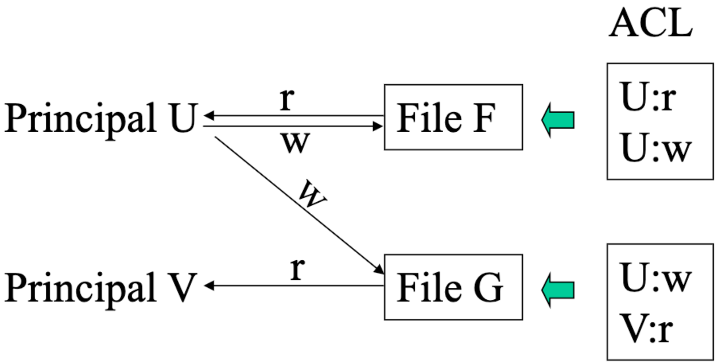
For special permissions or advanced settings, click Advanced.

3.1.3. Trojan Horse problem in DAC.

However, DAC does not provide real assurance - access restrictions can be easily bypassed with Trojan horse attack.



0. Initial state of Object / Subject / Rights:



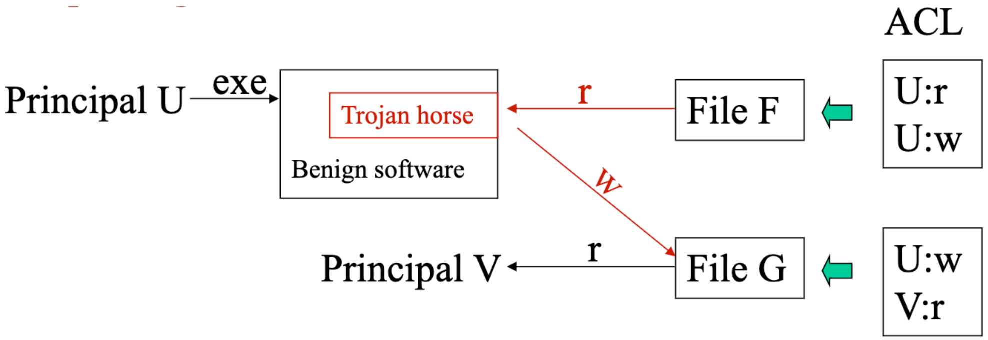
1. Principal V is an enemy who wants to read the secret file F.

2. Principal V sends U a benign software (.exe) with Trojan horse embedded.

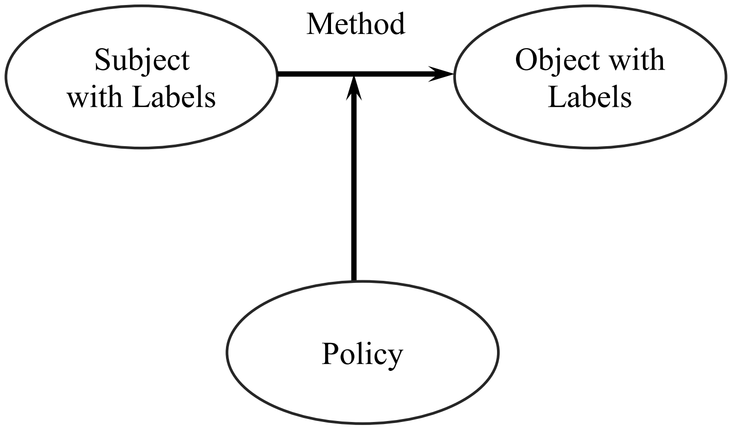
3. U executes the software and Trojan horse gains U's privileges.

4. Trojan horse make a copy data from F to G.

5. After attack principal V can now read file F secret.



3.2. MAC - Mandatory Access Control.



Since 1970.

- First formal describe in BLP model – 1974.
- Use in Orange Book US DoD standard – 1983.
- Use in Common Criteria international standard – 2005.
- Use for information confidentiality.

MAC use centralized Access Policy and based on security labels (mandates) attached to subjects and objects.

As a result, the creator (owner) of the object cannot change the access rights to this object, these rights are set by the security policy.

3.2.1. Implementation.

- SELinux added a MAC architecture to the Linux Kernel, which was merged into the mainline version of Linux in August 2003.
- FreeBSD and MacOS X implemented MAC as part of the TrustedBSD project.
- Microsoft since Windows Vista and Server 2008 adds Integrity Levels (Low, Medium, High, System, and Trusted Installer) to processes running in a login session.

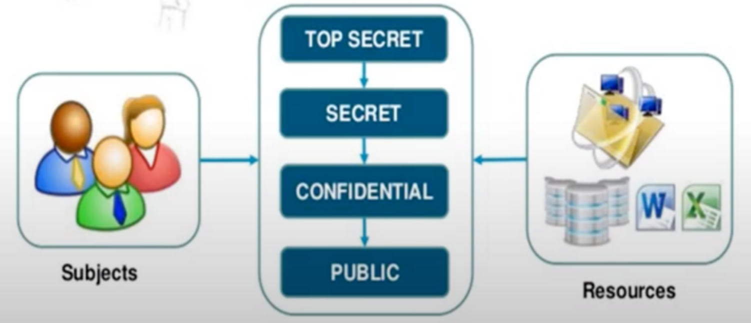
BLP model is applied to subjects, not to users, users are trusted, bet Subjects are not trusted because a Trojan horses attack is possible.

3.2.2. Bell-LaPadula Confidentiality Model.

- Published in 1974 by David Bell & Leonardo LaPadula (MITRE).
- A formal mathematical describe MAC Model.
- Only applies to confidentiality of information.

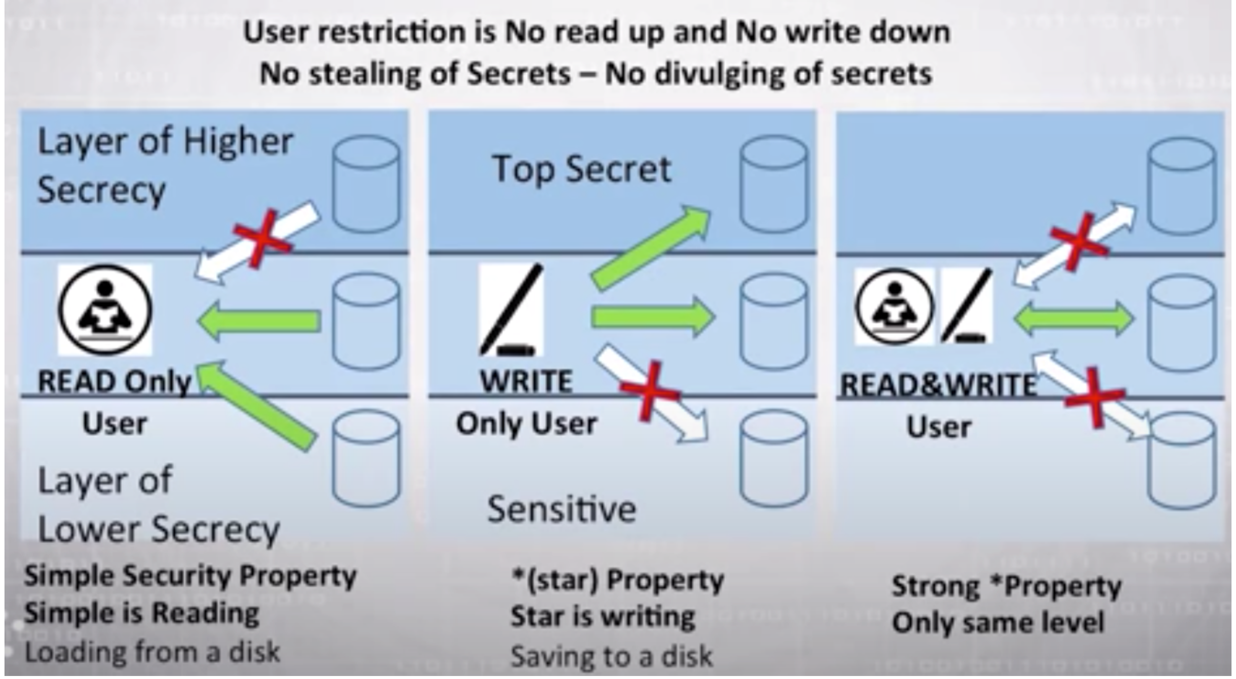
Every Objects and Subjects must be labeled as

- Top Secret (country military information)
- Secret (organizations bank detail access)
- Confidential (organizations business transactions)
- Public (organizations product list)
- Unclassified



The BLP model defines 3 mandatory access control (MAC) rules:

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) Security Property states that a subject at a given security level may not write to any object at a lower security level.
3. The Strong Star Property states that a subject at a given security level may not read/write to any object at a higher/lower security level.


User restriction is No read up and No write down
No stealing of Secrets – No divulging of secrets

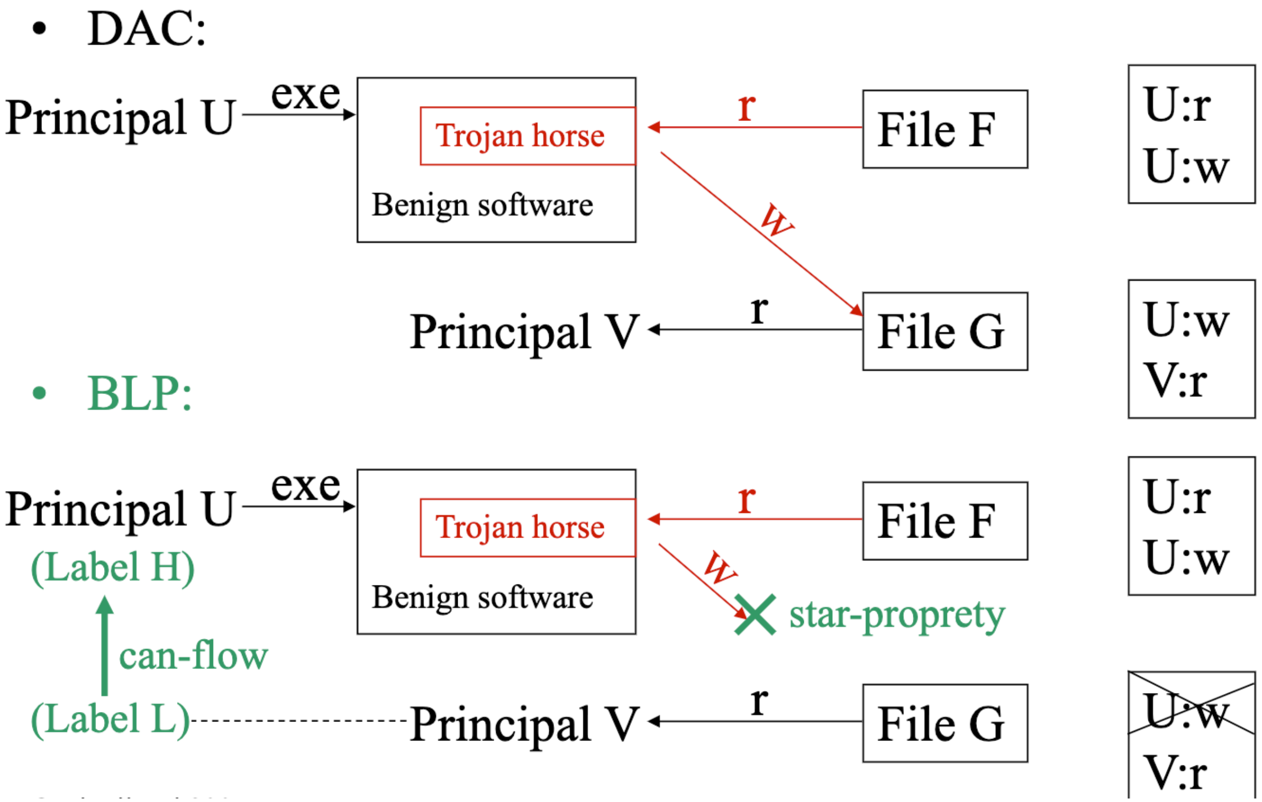
Simple Security Property
Simple is Reading
Loading from a disk

*(star) Property
Star is writing
Saving to a disk

Strong *Property
Only same level

3.2.3. Absence of a Trojan Horse problem in MAC (BLP).

Star-property prevents information leakage caused by Trojan horses



- DAC:

Principal U —exe→ Trojan horse / Benign software

File F —r→ Trojan horse

Trojan horse —w→ File G

File G —r→ Principal V

U:r
U:w

U:w
V:r

- BLP:

Principal U (Label H) —exe→ Trojan horse / Benign software

(Label L) —can-flow→ (Label H)

(Label L) - - - Principal V

File F —r→ Trojan horse

Trojan horse —w→ ✗ star-proprety

File G —r→ Principal V

U:r
U:w

~~U:w~~
V:r

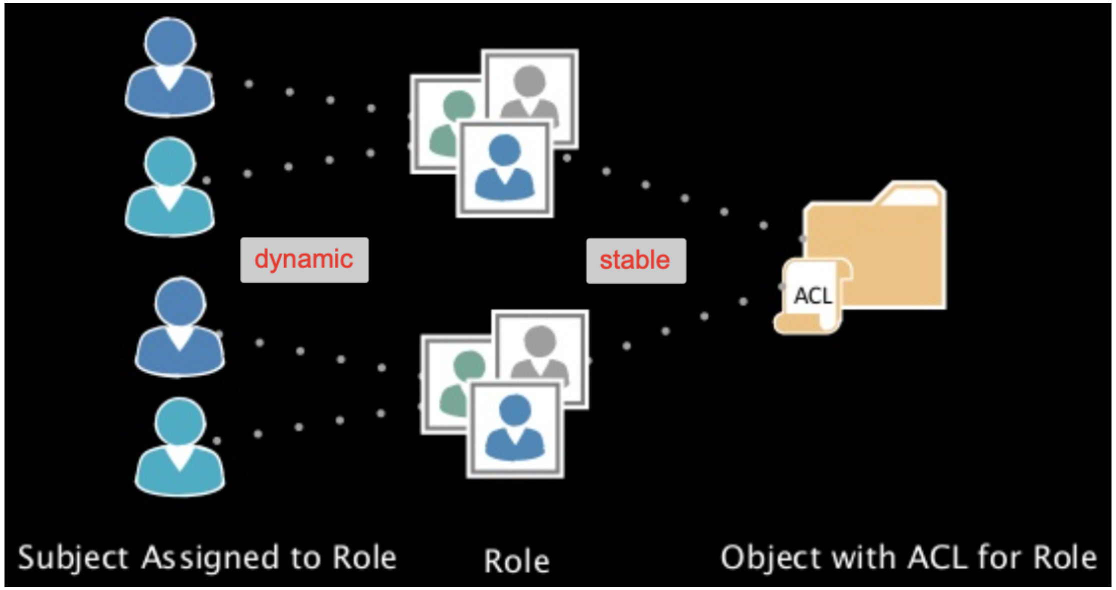
3.3. RBAC – Role Based Access Control.

Since 2001, have a INCITS 359-2012 Standard.

3.3.1. RBAC Concepts.

RBAC employs pre-defined roles that carry a specific set of privileges associated with them and to which subjects are assigned. For example, the role Manager and the role of Analyst.

RBAC specification made central management of enterprise access control capabilities possible and reduced the need for ACLs (ACLs minimization).



Role represents subject

- Specific task competency
- Job responsibility
- Specific duty assignment

Role defines permissions

- Student role in eLearn
- Professor role
- Admin and security officer role

Security management is simpler with roles

- Subject-role relationship dynamic changes over time
- Role-permission relationship is relatively stable

Implementation: Drupal**,** SRM, Microsoft Active Directory, Castl ALT Linux, SELinux, DBMS Oracle, Solaris 8.

DAC defines atomic rights to an object, and RBAC defines complex rights (privileges) to a system.

Using RBAC, both DAC and MAC can be implemented.

Session is a subset of the enabled roles for an actor.

Role is a subset of the rules describing the privileges for the available actions in the system.

Rules can be access and deny.

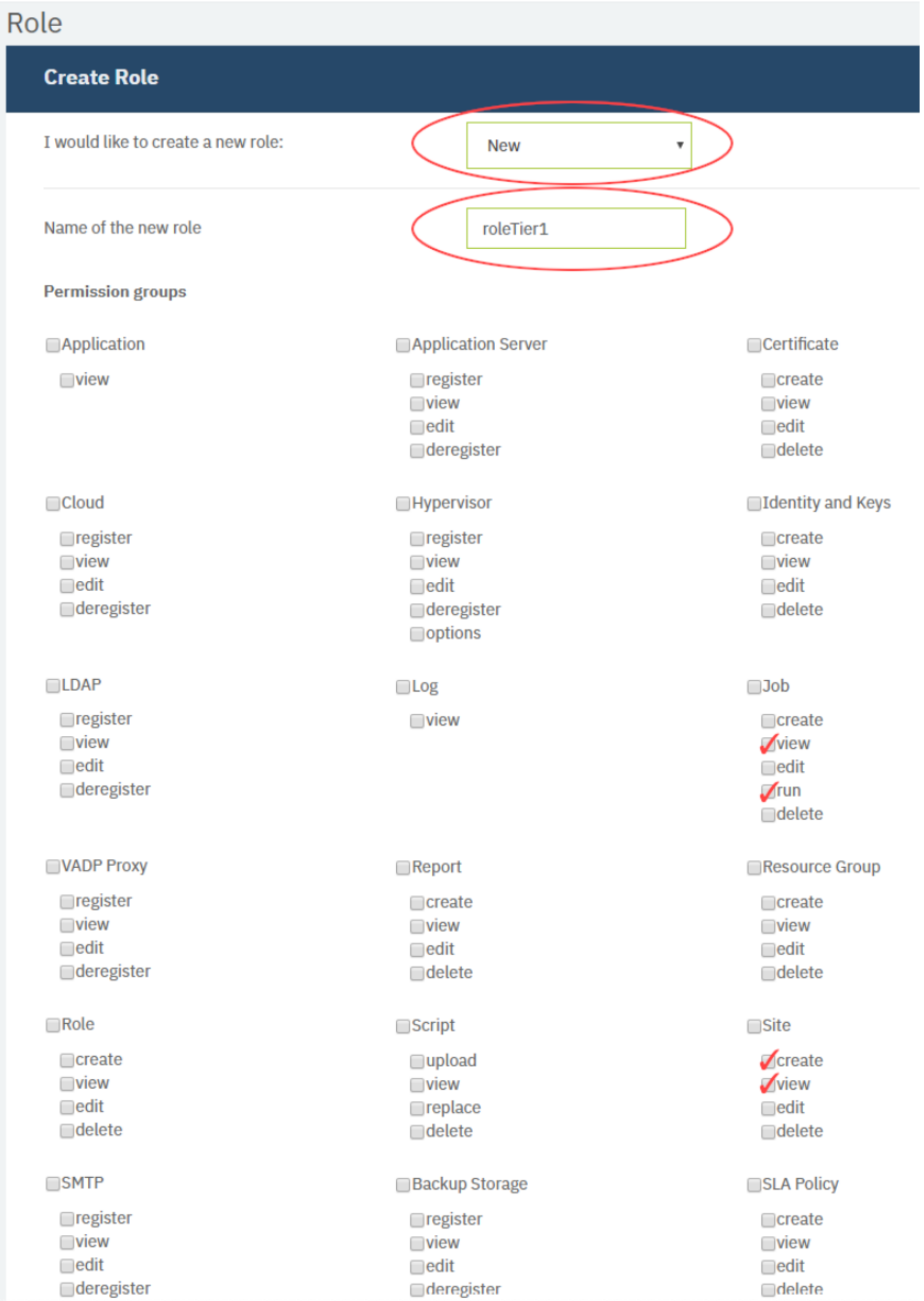
Role conflict - assignment of roles to one user with opposite sign rules (example, access read and deny read).

Role conflict resolution:

- prohibition of using rules with different signs;
- division of the subject into several different logins with conflict-free sets of roles;
- dynamic switching of sessions with the exclusion of a particular conflicting role.

Separation of roles, for example, static separation: the role of the System Administrator and the role of the Role Constructor are not combined; dynamic separation: two subjects with the role of the System Administrator are not possible at the same time.

Difference between group and role: S1 and S2 are both members of the Administrators group, but only one of them can have the Administrator role at the moment.



Role

Create Role

I would like to create a new role: New ▾

Name of the new role: roleTier1

Permission groups

□Application
□view

□Application Server
□register
□view
□edit
□deregister

□Certificate
□create
□view
□edit
□delete

□Cloud
□register
□view
□edit
□deregister

□Hypervisor
□register
□view
□edit
□deregister
□options

□Identity and Keys
□create
□view
□edit
□delete

□LDAP
□register
□view
□edit
□deregister

□Log
□view

□Job
□create
☑view
□edit
☑run
□delete

□VADP Proxy
□register
□view
□edit
□deregister

□Report
□create
□view
□edit
□delete

□Resource Group
□create
□view
□edit
□delete

□Role
□create
□view
□edit
□delete

□Script
□upload
□view
□replace
□delete

□Site
☑create
☑view
□edit
□delete

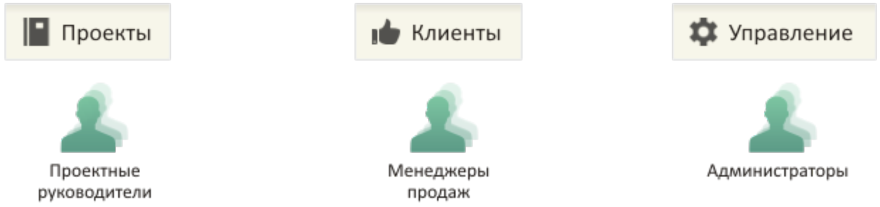
□SMTP
□register
□view
□edit
□deregister

□Backup Storage
□register
□view
□edit
□deregister

□SLA Policy
□create
□view
□edit
□delete

3.3.2. RBAC Disadvantages.

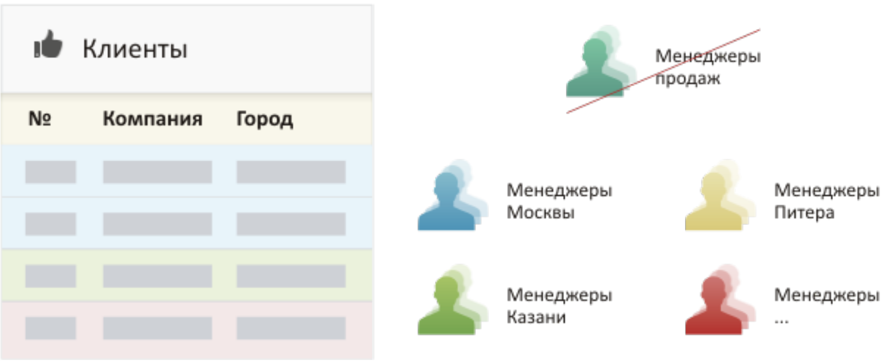
For example, in System have three resource groups and three roles. Systems can configured by Administrators, Clients is available from the Sales Department, Projects are managed by Project Managers.



This system is convenient and simple, but in large companies it turns into a real hell (ад).

Problem 1. To solve the problem ***"Managers see clients only from their city",*** you will have to create many ephemeral roles: Moscow managers, Peters managers, Kazan managers, and so on.

Problem 2. The number of roles grows many times over with the addition of one more characteristic.



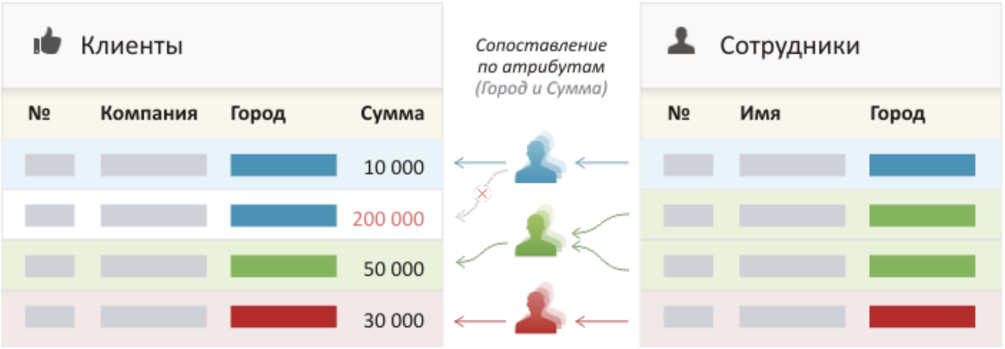
For example, what to do if the task is formulated as follows:
“Junior managers see the orders of their branch if the order amount is less than 100 000 EUR”?

It becomes impossible to manage this !

RBAC problem Solution.

There is an alternative to Role-based policy - Attribute-based access control (**ABAC**) policy.

An attribute can be any property (field) of a data object or employee.

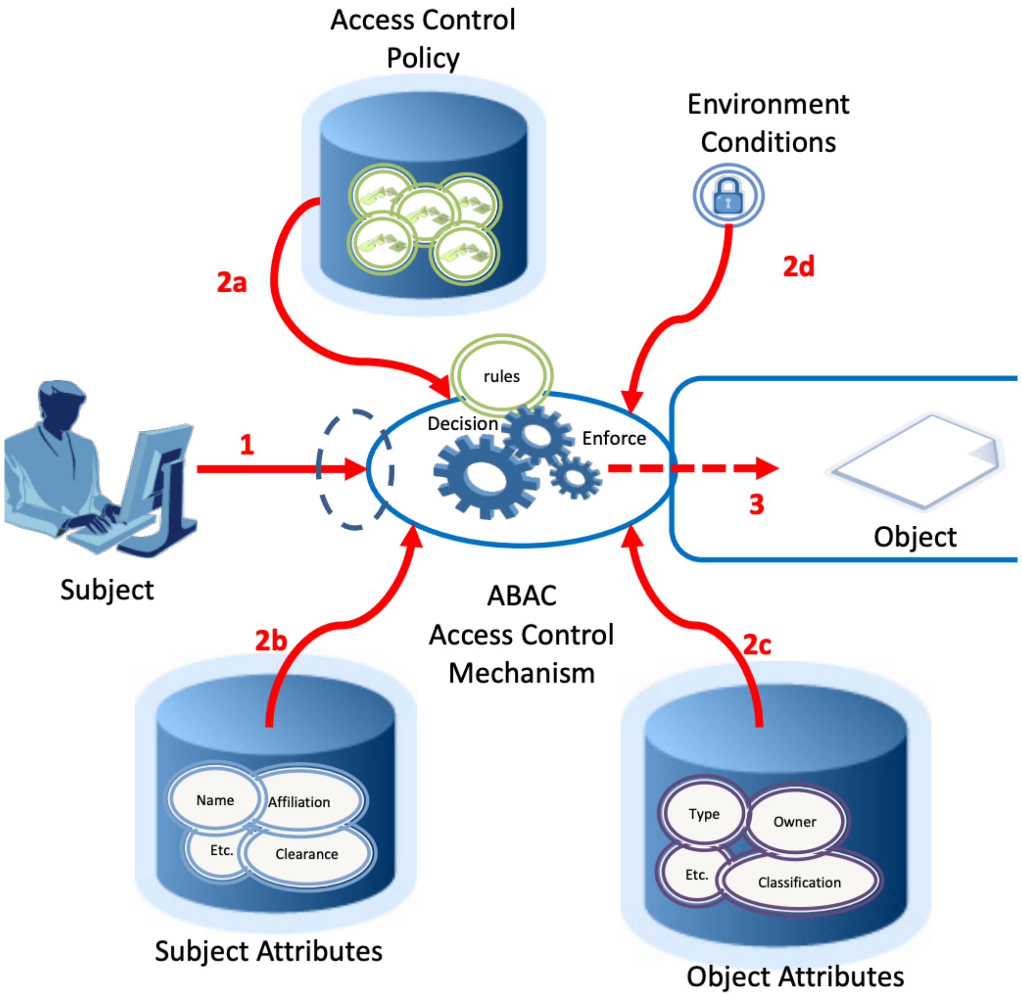


For example, "Order amount > 100 000 EUR" or "Customer city = Employee city".

3.4. ABAC.

Describe NIST800-162.

ABAC is the most flexible mechanism to implement authorization.



3.4.1. Basic ABAC Scenario.

1. Subject requests access to object.

2. Access Control Mechanism evaluates

    a) Rules,
    b) Subject Attributes,
    c) Object Attributes,
    d) Environment Conditions (Attributes)

    to compute a Decision (verdict).

3. Subject is given access to object if authorized verdict set.

3.4.2. Attributes.

Attributes can be about anything and anyone.

1. Subject attributes: attributes that describe the user attempting the access e.g. name, age, clearance, department, role, gender, job title...

2. Action attributes: attributes that describe the action being attempted e.g. read, delete, view, approve...
3. Object attributes: attributes that describe the object (or resource) being accessed e.g. the object type (medical record, bank account...), the department, the classification or sensitivity, the location...
4. Contextual (environment) attributes: attributes that deal with time, location or dynamic aspects of the access control scenario, shielded phone, ...

3.4.3. Implementation.

1. Concept of ABAC can be used at the firewall, server, application, database, and data layer.

2. One standard that implements attribute- and policy-based access control is XACML, the eXtensible Access Control Markup Language.

3. XACML defines only an architecture, a policy language, and a request / response scheme.

4. Attribute and access management execute which traditional tools, databases, and directories.

5. Fedora 3.3 FESL (Fedora Security Layer) using XACML to implement ABAC.

6. Companies, for example, the United States governmental and military agencies, have started using a basic level of ABAC - protects data with ‘IF/THEN/AND’ rules for dynamic access data to users.

7. The use of attributes bring additional context to grant or deny access. Attribute and access management execute which traditional tools, databases, and directories.

8. As of Windows Server 2012, Microsoft has implemented an ABAC approach to controlling access to files and folders.

3.5. Other Access Control Models.

3.5.1. Biba Model.

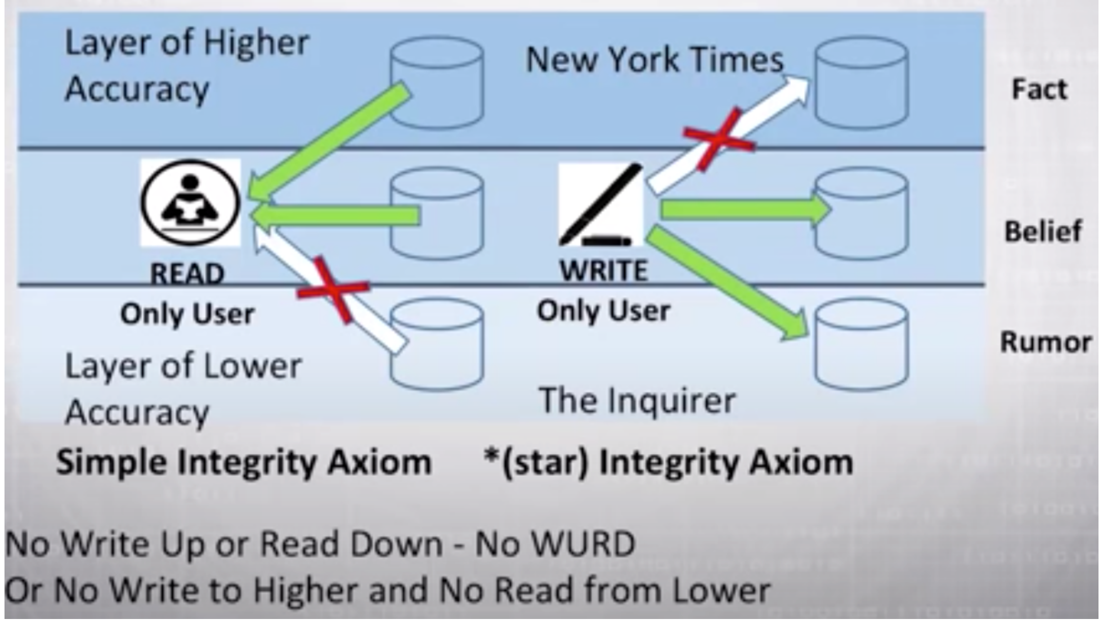
Published in 1977

Biba Integrity model describes a set of access control rules that are designed to ensure data integrity. Subjects and Objects are grouped into various ordered levels of integrity.

Access modes of Biba Model

- **Modify**: This allows a subject to write to an object. In layman parlance, it is equivalent to write mode in other models.
- **Observe**: This allows a subject to read an object. This command is synonymous to the read command of other models.
- **Invoke**: This allows one subject to communicate with another subject.
- **Execute**: This allows a subject to execute an object. The command essentially allows a subject to execute a program which is the object

This model is directed towards data integrity (rather than confidentiality). It is also called **“read up, write down”** model. This implies users can only write content at or below their own integrity level. Similarly, users can only read content at or above their own integrity level.



Set of rules defined by Biba model:

1. The **Simple Integrity Property** states that a subject at a given level of integrity must **not read data at a lower integrity level** (no read down)

Mathematically, denoted as: ***s ∈S can observe o∈O if and only if i(s) ≤ i(o)***

2. The *** (star) Integrity Property** states that a subject at a given level of integrity must **not write data at a higher level of integrity** (no write up).

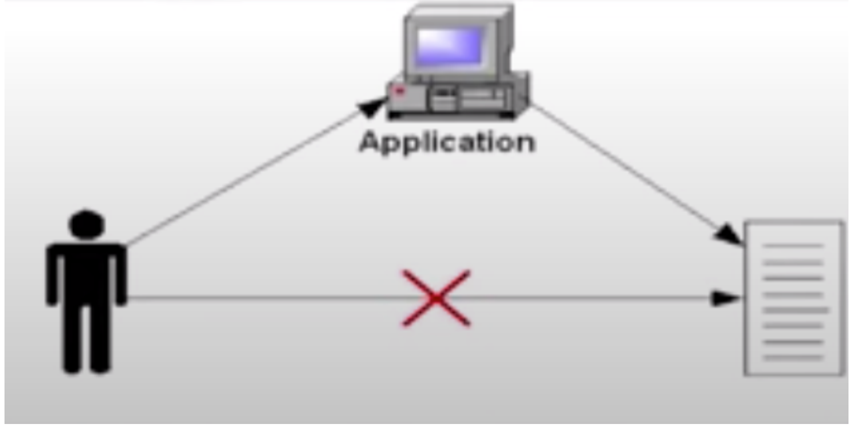
Mathematically, denoted as: s ∈S can modify to o∈O if and only if i(o) ≤ i(s)

Invocation Property states that a **process from below cannot request higher access**; only with subjects at an equal or lower level.

Mathematically, denoted as: $s_1 \in S$ *can invoke* $s_2 \in S$ *if and only if* $i(s_2) \leq i(s_1)$

3.5.2. Clark - Wilson Model.

Use for commercial integrity – base on rule of least privilege and non-DAC – transaction with second subject);



Review

3.5.3. Take-Grant Model.

By the Lipton and Snyder in 1976.

It is a formal model used in the field of computer security to analyze DAC systems.

The state of the system is described by its graph G.

G = (S, O, R) – finite, labeled, directed graph without loops;

● – S (subjects set active entities like users, processes, …);

o – O (objects set passive entities like files, memory segments, ...);

⊗ – don't care (either a subject or an object);

R ∈ {r,w,x,…}U{t,g} – set of rights, arcs of the graph;

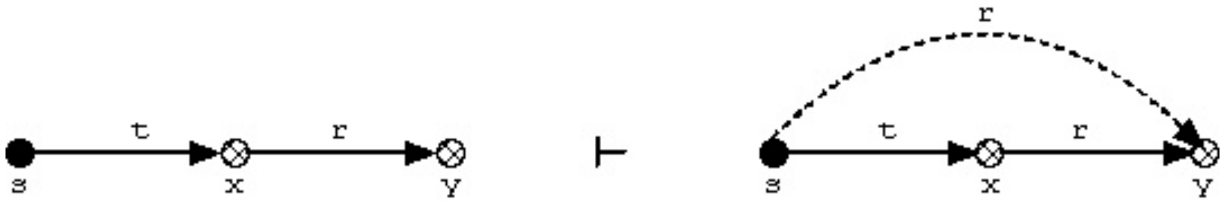
t – the right to take "access rights";

g – the right to grant "access rights".

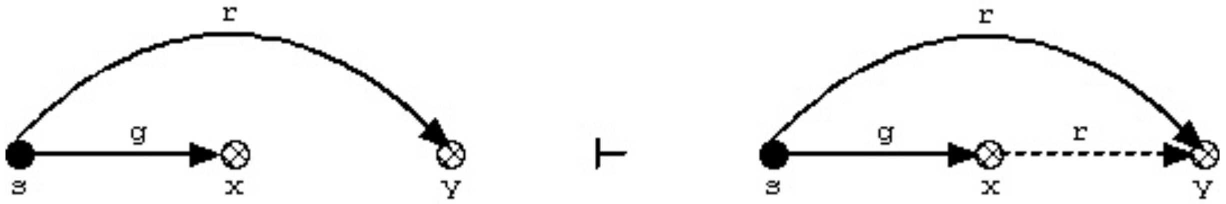
Rules – convert G to graph G'.

State Transition Rules.

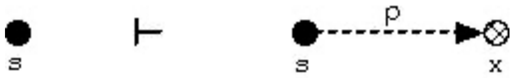
Rule 1 (Take): Let *s* be a subject with *t* being an element of (*s*,*x*) and *r* being an element of (*x*,*y*) for a right *r* and the nodes *x*, *y*. To add *r* to (*s*,*y*) use: *s* **take** *r* for *y* from *x*; where nodes *x* and *y* can be either subjects or objects.



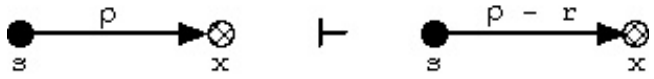
Rule 2 (Grant): Let *s* be a subject with *g* being an element of (*s*,*x*) and *r* being an element of (*s*,*y*) for a right *r* and the nodes *x*, *y*. To add *r* to (*x*,*y*) use: *s* **grant** *r* for *y* to *x*; where nodes *x* and *y* can be either subjects or objects.



Rule 3 (Create): If *s* is a subject and *p* is a set of rights, then the command: *s* **create** *p* for **new** {**subject** or **object**} *x* will add a new node *x* and sets (*s*,*x*) = *p*; where node *x* can be either a subject or an object.

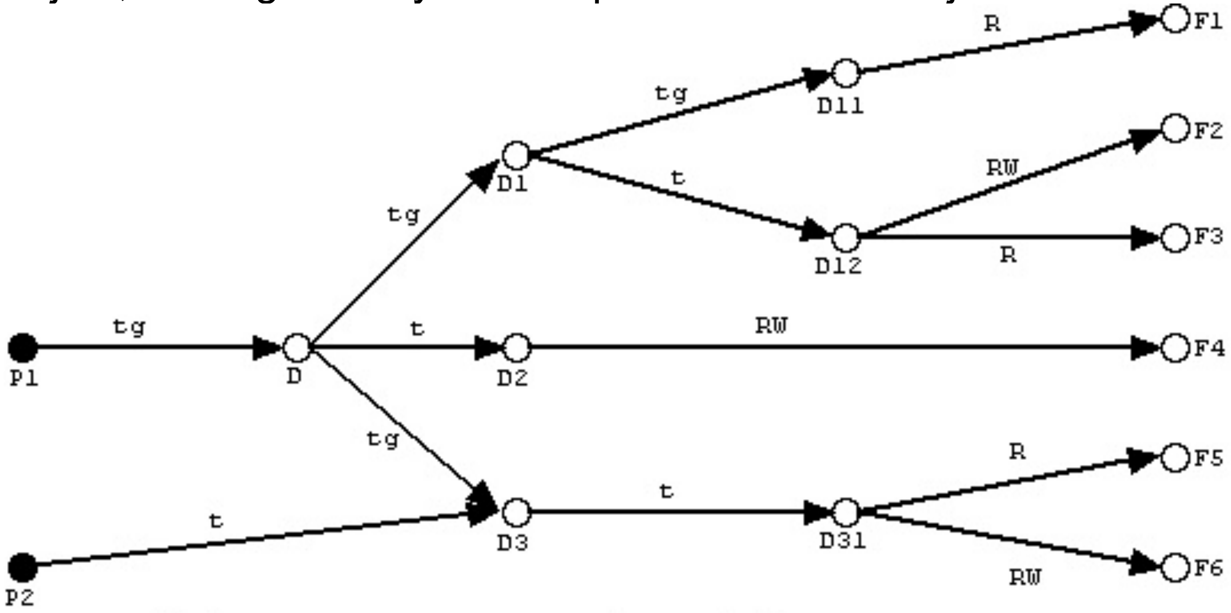


Rule 4 (Remove): If *s* is a subject and *x* is a node, then the command: *s* **remove** *r* for *x* will remove the right *r* from (*s*,*x*); where node *x* can be either a subject or an object.



Example 1.

The figure below is a graphical representation of a directory structure. In this graph **P1** and **P2** represent subjects (possible users), and the **D**s and **F**s represent objects, directories and files respectively. *Read* rights have been changed to *take* rights for all levels except for the actually file level in the directories. *Write* rights have equally been changed to *grant* rights.
It becomes clear from this graph that if a subject/object has read (take) capability on an object, it can take on any of the capabilities that this object has. Similarly, if a subject/object has write (grant) capabilities to an object, it can grant any of its capabilities to that object.

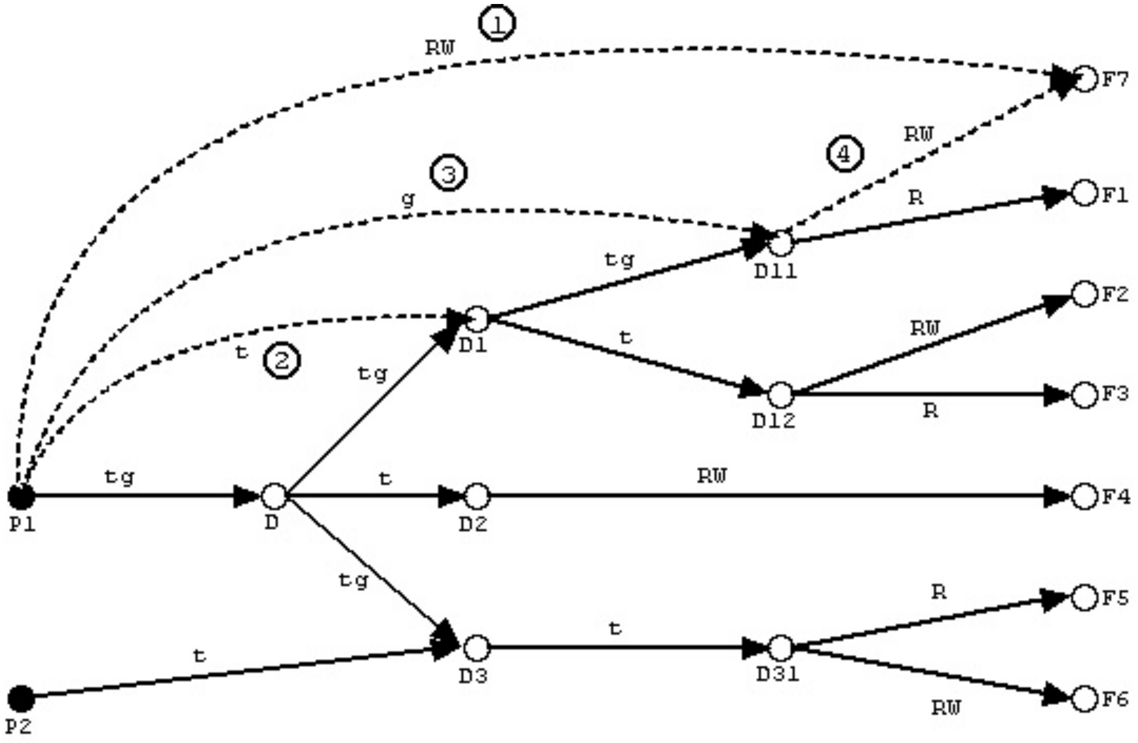


Take-grant representation of directory structure

Example 2.

The figure below shows that through combinations of the above four rules, a new file can be added to the directory structure of Example 1 and Read/Write rights issued for the directory in which the file exists. The following four steps are necessary to add the file and grant rights:

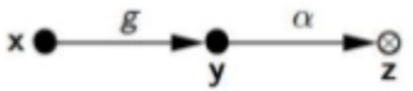
1. *P1* **create** *RW* for **new object** *F7*
2. *P1* **take** *t* for *D1* from *D*
3. *P1* **take** *g* for *D11* from *D1*
4. *P1* **grant** *RW* for *F7* to *D11*



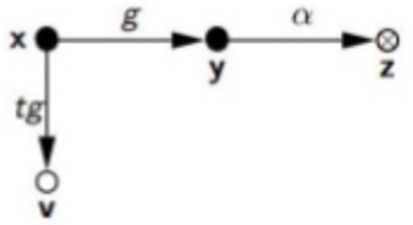
Adding a new file F7 to directory D11

Example 3.

1.

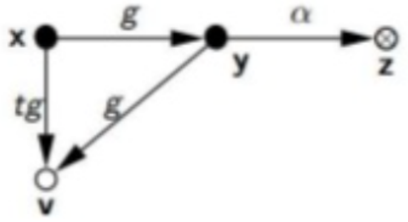


2.



x creates (tg to) new v

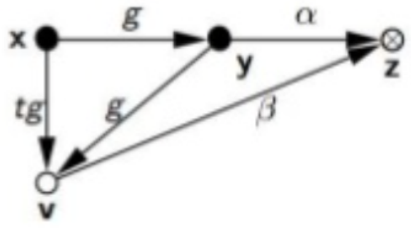
3.



x creates (tg to) new v

x grants (g to v) to y

4.

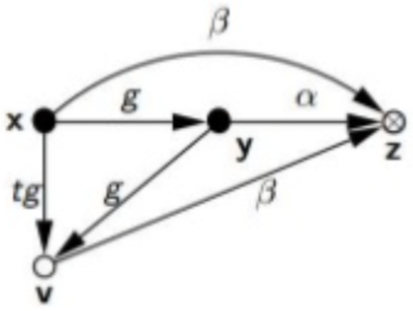


x creates (tg to) new v

x grants (g to v) to y

y grants (β to z) to v

5.



x creates (tg to) new v

x grants (g to v) to y

y grants (β to z) to v

x takes (β to z) from v

3.6. CISSP Practice Questions.

CISSP - Certified Information Systems Security Professional.

3.6.1. Which of the following provides the most flexible access control?

A. A subject asserting unmarried
B. A subject with the Top Secret clearance
C. A subject with need-to-know
D. A subject assigned to the Admin role

The recommended answer is A, A subject asserting unmarried. This question is designed to help you understand the characteristics of the common access control mechanisms as follows:

A ⇒ ABAC; B ⇒ MAC; C ⇒ DAC; D ⇒ RBAC.

3.6.2. Which of the following is subject to Trojan Horse attack?

A. DAC
B. MAC
C. RBAC