System Security

Waltzing Access Control in Unix 2/2

Mohamed Sabt

Univ Rennes, CNRS, IRISA

2023 / 2024

Part III

DAC in Theory
(Discretionary Access Control)

Discretionary Access Control

- User-oriented security policy (based on ID of requestor).
- DAC requires subjects to be authenticated before access to a particular object!

- **Discretionary** because an entity has rights to enable another entity to access a resource.
- General approach as used in operating systems and database management systems is that of an **access matrix**.
  - Lists of subjects in one dimension (rows).
  - Lists of objects in the other dimension (columns).
  - Each matrix entry specifies access rights of the specified subject to that object.

Access Matrix: Example

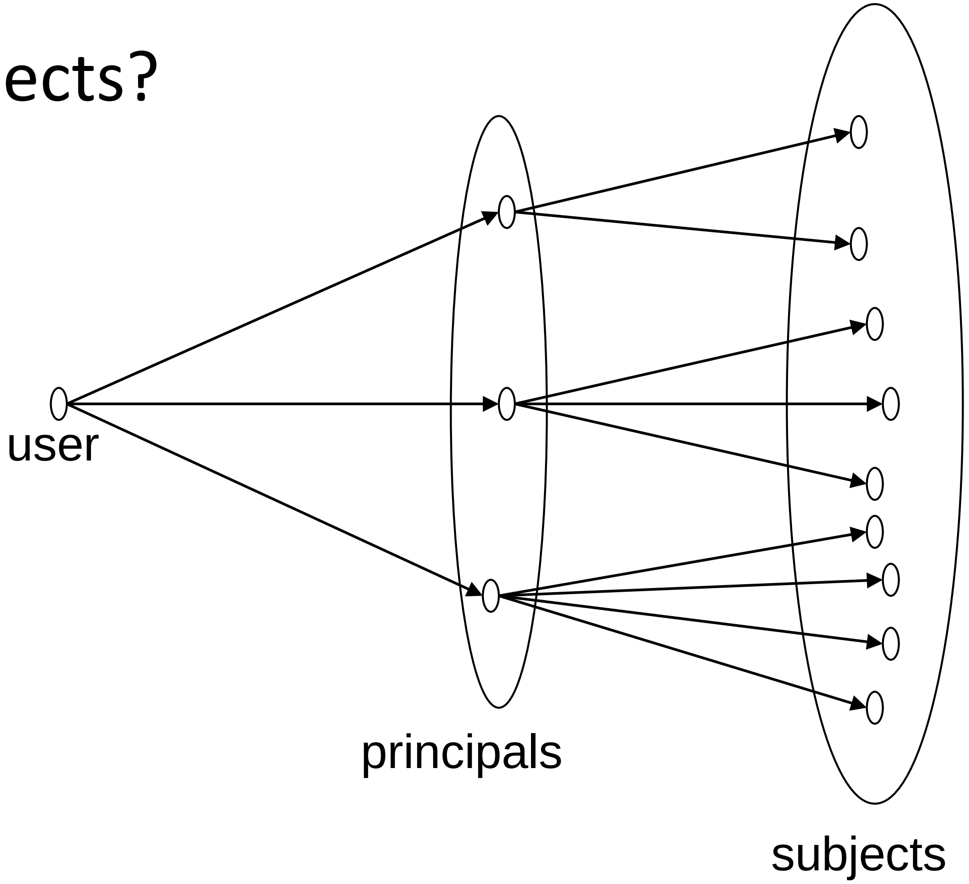
Objects

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

Access rights

Access Matrix Element: Subjects

- Relationship between Users and Subjects?
  - User – a real world user.
  - Principal – a unit of access control
  and authorization.

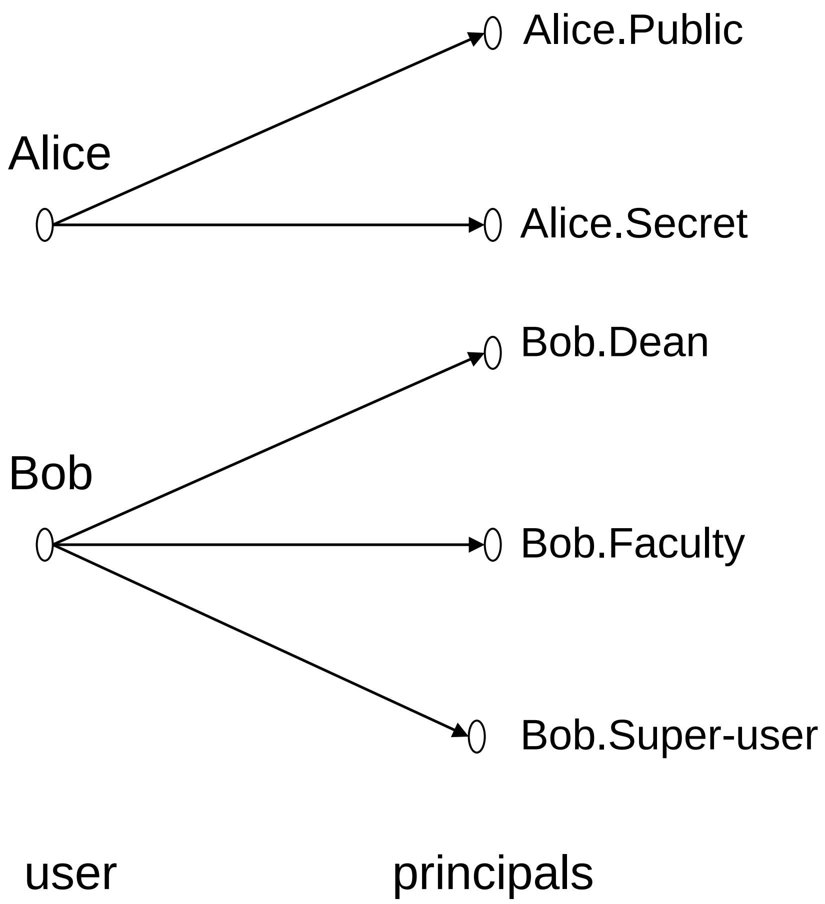


User -- Principal

- One to many mapping between user and principals.

- System authenticates user in the context of principal.

- Shared principals (accounts) are not good for accountability.

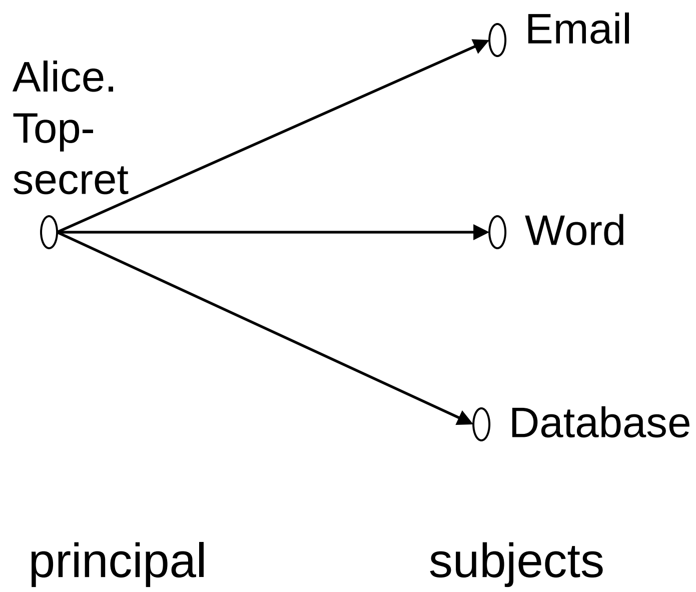


Principal -- Subject

- One to many mapping between principal and subjects.

- A subject is a program or application run on behalf of principal.

- Subjects are often treated the same as principal if all subjects of a principal have the same rights.



Access Matrix Element: Objects

- An object is anything on which a subject can perform operations (mediated by access rights).

- Usually objects are passive, for example
  - File
  - Directory
  - Memory segment
- But, subjects can also be objects, with operations
  - Kill
  - Suspend
  - Resume

Access Matrix Elements: Rights

- A right specifies what kind of access a subject can perform on an object
  - Read
  - Write
  - Execute
  - Create
  - Delete
  - Transfer
  - …

Implementation of an Access Matrix

- In practice, an access matrix is usually sparse.
- Each column of access control matrix is stored with the corresponding Object.

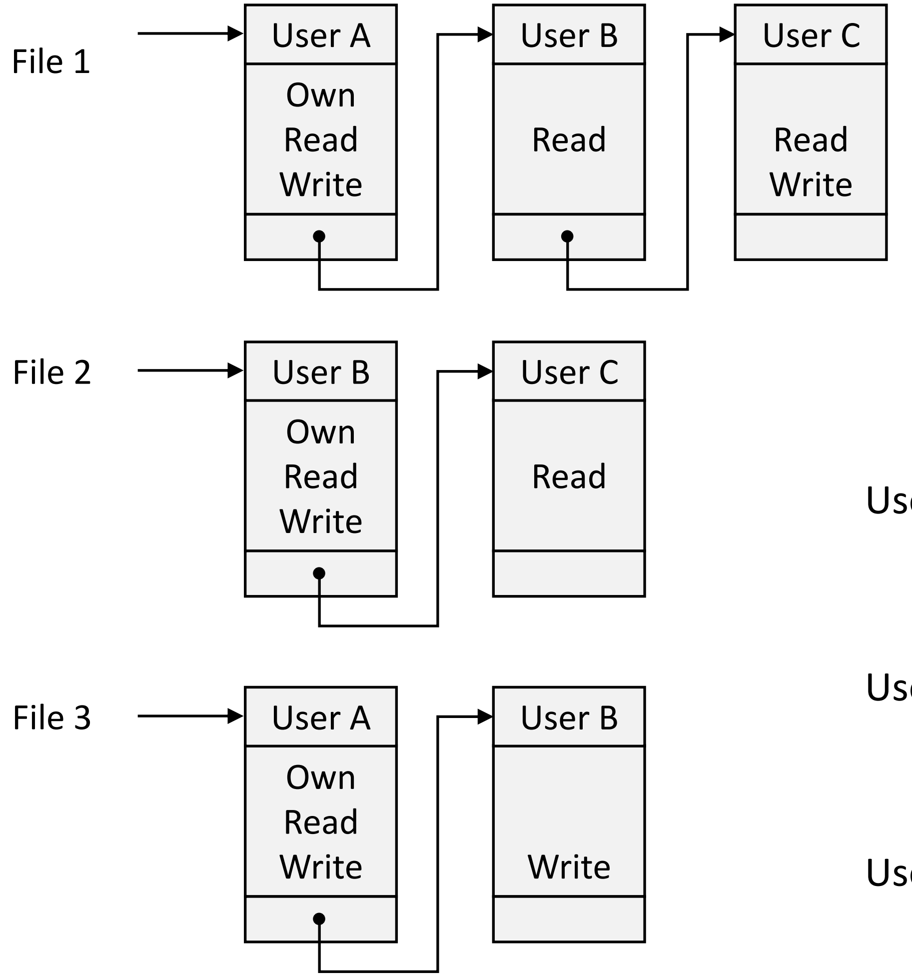
- Therefore implemented by decomposition in one of two ways
  - By columns – access control lists
  - By rows – ?

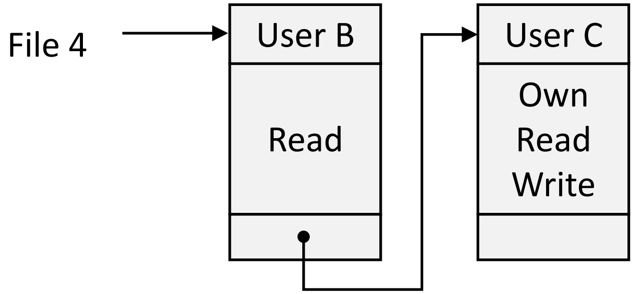
	File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

Access Control Lists (ACL)

- Access rights stored with objects.
- ACL may contain default (public) entries
  - If user not explicitly listed in ACL – default rights (e.g., read only).
  - Elements of ACL include individual users as well as groups of users.
- ACLs are convenient when desired to determine which subjects have which access rights to particular resource
  - Not convenient for determining the access rights of a particular user.

Access Matrix Vs ACL

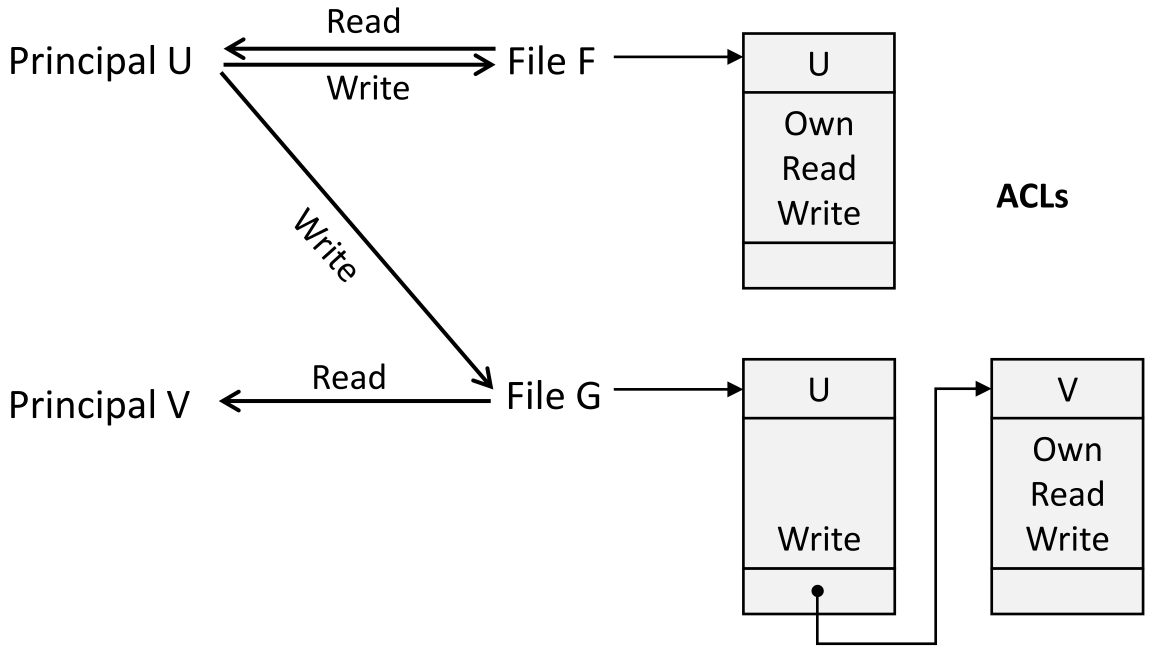




	File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

Security Problems of DAC 1/2

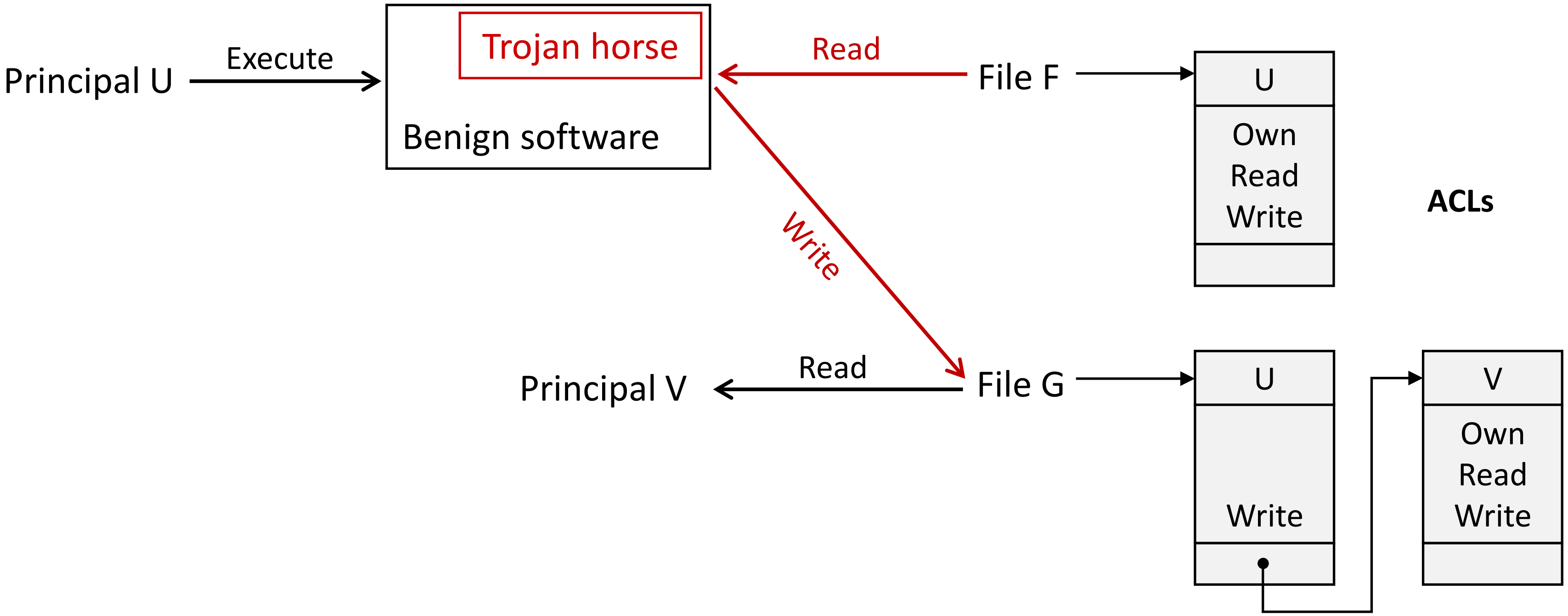
- However, DAC does not provide real assurance – access restrictions can be easily bypassed.
  - Trojan horse attack.



Principal V is a bad guy who wants to read file F.

Security Problems of DAC 2/2

- Principal V sends U a benign software with Trojan horse.
- U executes the software → Trojan horse gains U's privileges.



Principal V is a bad guy who can read the file F with the help of Trojan horse

Solution to the DAC Security

- Mandatory Access Control (MAC)

Part IV

MAC in Theory (Mandatory Access Control)

Mandatory Access Control (MAC)

- A MAC policy is a means of assigning access rights based on regulation by a central authority.

- The philosophy underlying these policies is that information belongs to an organization (rather than individual members of it).
  - Organizations should control the security policy.

- MAC policies strive to defend against Trojan horse attacks.

Multi-Level Security(MLS)

- MAC attaches security labels to subjects and objects
  - Security label to subject → security clearance
  - Security label to object → security classification
- Security Classification:
  - Military: Unclassified (anyone can see this), confidential, secret, and top secret.
  - Business: public, sensitive, proprietary, and restricted.
- System controls access to resources by comparing security labels of the resources (e.g. system, high, low security) with security clearances of subjects accessing the resources.
- Users have no control of security labels (in contrast to DAC)
  - Note that cleared entity cannot pass on access rights to another entity (as is the case in DAC)
  - Denying users full control over the access to resources that they create.

Bell-LaPadula (BLP)

- BLP is a model for achieving MLS
  - Introduced in 1973.
- Main objective:
  - Enable one to formally show that a computer system can securely process classified information.

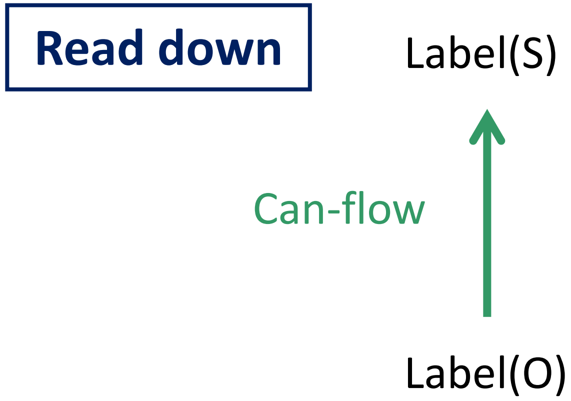
- Each subject has a current security level.
- Each object has a classification level.
- A computer system is modeled as a state-transition system.
  - Each state has objects, and the current access information.
  - There are state transition rules describing how a system can go from one state to another.

BLP Model 1/2

Simple-security property:

Subject S can read Object O only if

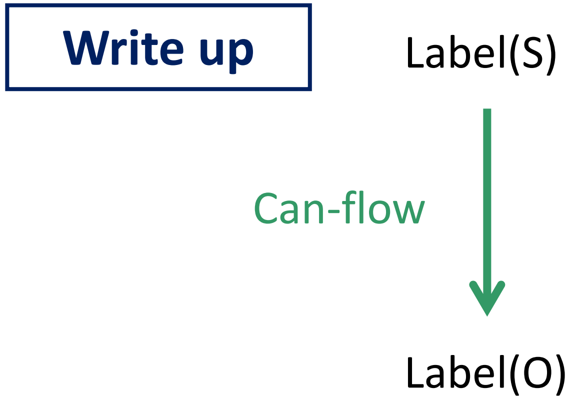
- Label(S) dominates Label(O)
- Information can flow from Label(O) to Label(S)



Star-property

Subject S can write object O only if

- Label(O) dominates Label(S)
- Information can flow from Label(S) to Label(O)



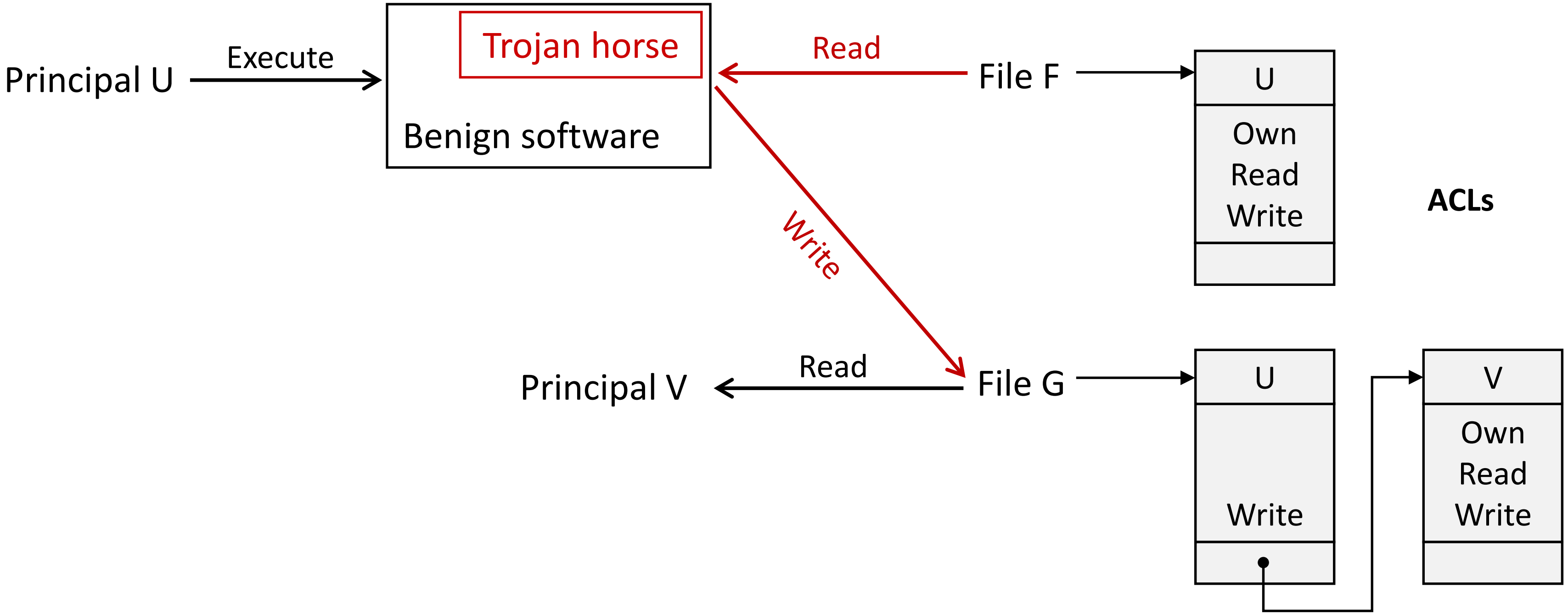
BLP Model 2/2

- BLP model is applied to subjects, not users.
  - Users are trusted.
  - Subjects are not trusted due to Trojan horse.

- Star-property prevents information leakage caused by Trojan horses.

Recall the Security Problem of DAC

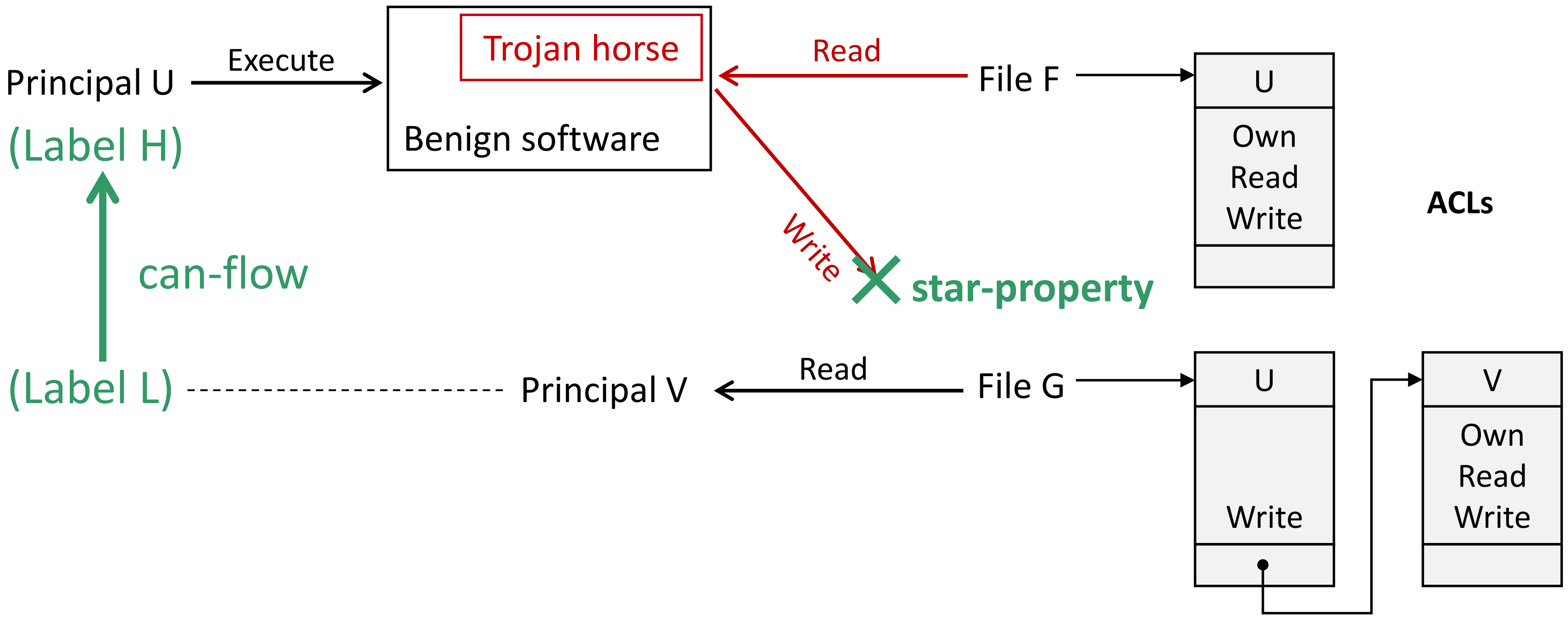
- Principal V sends U a benign software with Trojan horse included.
- U executes the software → Trojan horse gains U's privilege.



- Principal V can read file F with the help of Trojan horse

BLP Star Property Solves the Problem

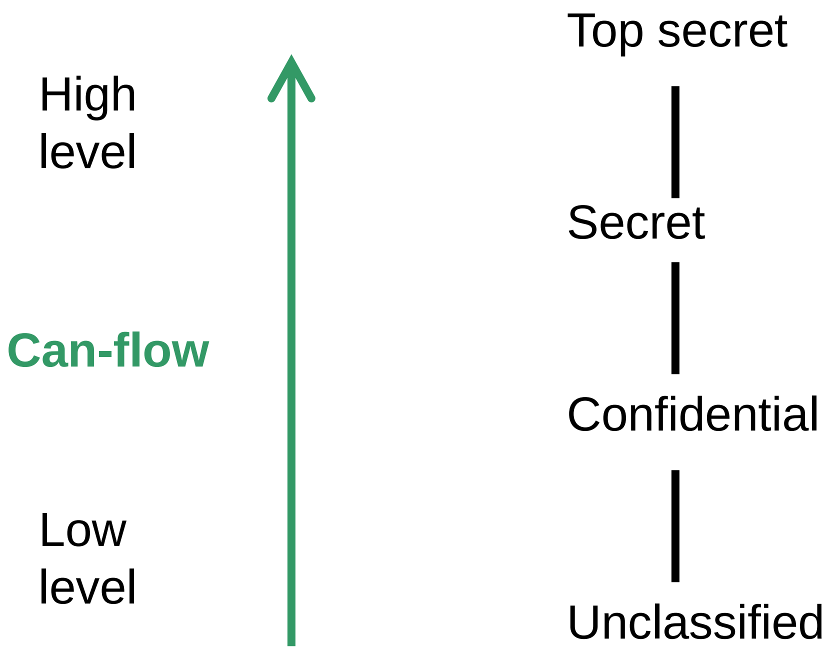
- Assign a high (sensitive) security label to Principal U and file F and low (public) security label to principal V and file G.



- Note that the star property overrides ACL access rights

Controlling Information Flow -- Confidentiality

- Military security classes as security labels



- If subject's level is equal to or greater than the object's level, the subject is allowed to read the object (**read down**).
- Note that a subject may only **write up**.

BLP Not Enough

- The BLP model is concerned with only confidentiality, not integrity.
  - For instance, subjects can “write up”. Thus, a subject that cannot read an object is permitted to make changes to that object (blind write).
  - Yes, it makes little sense to trust a subject to modify the information contained in an object, if that subject is not trusted to read the information contained in the object.
- A Subject must login at a lower level than their clearance if they want to “write down” (e.g. unclassified document).
  - This is annoying for users, but also a nice application of the Principle of Least Privilege.
- Some processes must be allowed to violate the BLP security conditions.
  - For instance, an encryption program takes secret information and outputs encrypted but unclassified information.
  - This program seems to violate the no “write down” condition.

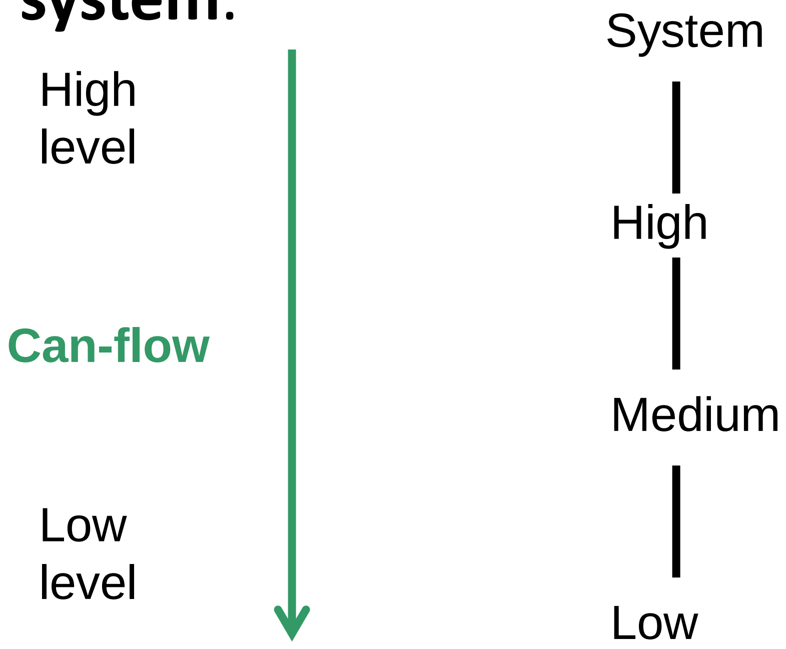
Biba: Integrity Levels

- Each object has an integrity level.

- Integrity levels are different from security levels in confidentiality protection
  - Highly sensitive data may have low integrity.

Controlling Information Flow -- Integrity

- Windows Mandatory Integrity Control (MIC) defines 4 integrity levels: **low**, **medium**, **high** and **system**.



- If subject's level is equal to or greater than the object's level, the subject is allowed to write to or delete the object (**write down**).
- Else, can only read if allowed by the ACL (**read up)**.

BLP vs. Biba 1/2

Confidentiality	Integrity
Control reading. Preserved if confidential info is not read.	Control writing. Preserved if important object is not changed.
For subjects who need to read, control writing after reading is sufficient.	For subjects who need to write, control reading before writing is not sufficient.

BLP vs. Biba 2/2

- For confidentiality, controlling reading & writing is sufficient
  - Theoretically, no subject needs to be trusted for confidentiality.

- For integrity, controlling reading and writing is not sufficient
  - One has to trust all subjects who can write to critical data.
  - How to establish trust in subjects becomes a challenge.

Clark-Wilson

- In 1987, Clark and Wilson defined their MAC model focusing on the integrity of data.
- Two high-level mechanisms for enforcing data integrity
  - Well-formed transaction
  - Separation of duty

- Well-formed transaction: a user should not manipulate data arbitrarily, but only in constrained ways that preserve or ensure data integrity.
  - E.g. making two sets of entries for everything that happens.
  - E.g. append-only log to record all transactions (mistakes are not erased).

- Separation of duty: transactions are separated into subparts that must be done by independent parties (supposing that there is no collusion between agents working into different subparts).
  - E.g. a person who can create or certify a well-formed transaction may not execute it.

Trusted Procedures (TPs)

- All subjects must be authenticated.
- All TPs (and the operations they perform) must be logged.
- All TPs must be approved by a central authority.
- No data may be changed except by a TP.
- All subjects must be cleared to perform particular TPs by a central authority.

- To Sum Up, use the AAA principle (Authentication, Audit and Authorization).

Clark-Wilson Mechanisms

- Control access to data: a data item can be manipulated only by a specific set of programs.
- Control access to programs: each user must be permitted to use only certain sets of programs.
- Program certification: programs must be inspected for proper construction, controls must be provided on the ability to install and modify these programs.
- Control administration: assignment of people to programs must be controlled and inspected.

Clark-Wilson Vs BLP

- A data item is not associated with a particular security level, but rather with a set of TPs.
- A user is not given read/write access to data items, but rather permissions to execute certain programs.

Clark-Wilson Vs Biba

- Biba lacks the procedures and requirements on identifying subjects as trusted.
- Clark-Wilson focuses on how to ensure that programs can be trusted.

MAC in Real Life 1/2

- Security-Enhanced Linux (SELinux)
  - Use Linux Security Module to implement MAC.
  - Enforce MAC policies that confine user programs and system servers to minimum amount of privilege they require to do their jobs.

- AppArmor ("Application Armor")
  - A security module for the Linux kernel.
  - Administrator can associate with each program a security profile that restricts the capabilities of that program.

MAC in Real Life 2/2

Mandatory Integrity Control in Windows (since Vista)

- Uses four integrity levels: Low, Medium, High and System.
- Each process is assigned a level, which limits resources it can access.
- Processes started by normal users have Medium.
- Elevated processes have High.
- Some processes run as Low, such as IE in protected mode.

Part V

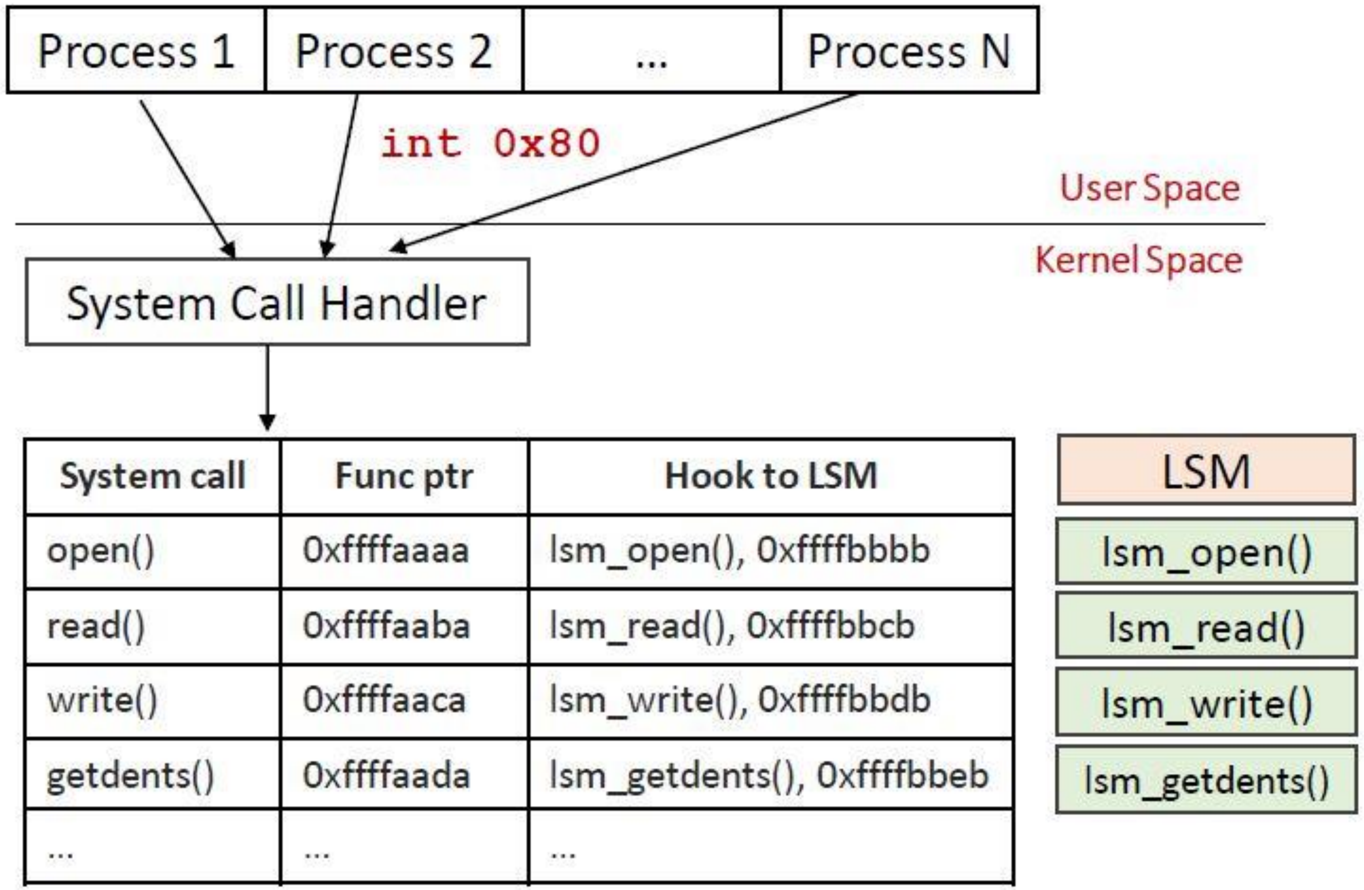
MAC in Practice
Security-Enhanced Linux

Linux Security Module (LSM)

- Implementation of a reference monitor.
  - A mechanism to enforce access control.
- Adopting LSM in Linux Kernel
  - Originally a set of kernel modules in 2.2, updated in 2.4.
  - LSM Feature in 2.6
    - SELinux developed by the NSA and released in 2001.
    - Default choice for Fedora/RedHat Linux
- LSM Design Principle
  - Independent of the implementation of the security policy.
  - Modifies as little as possible in the Kernel

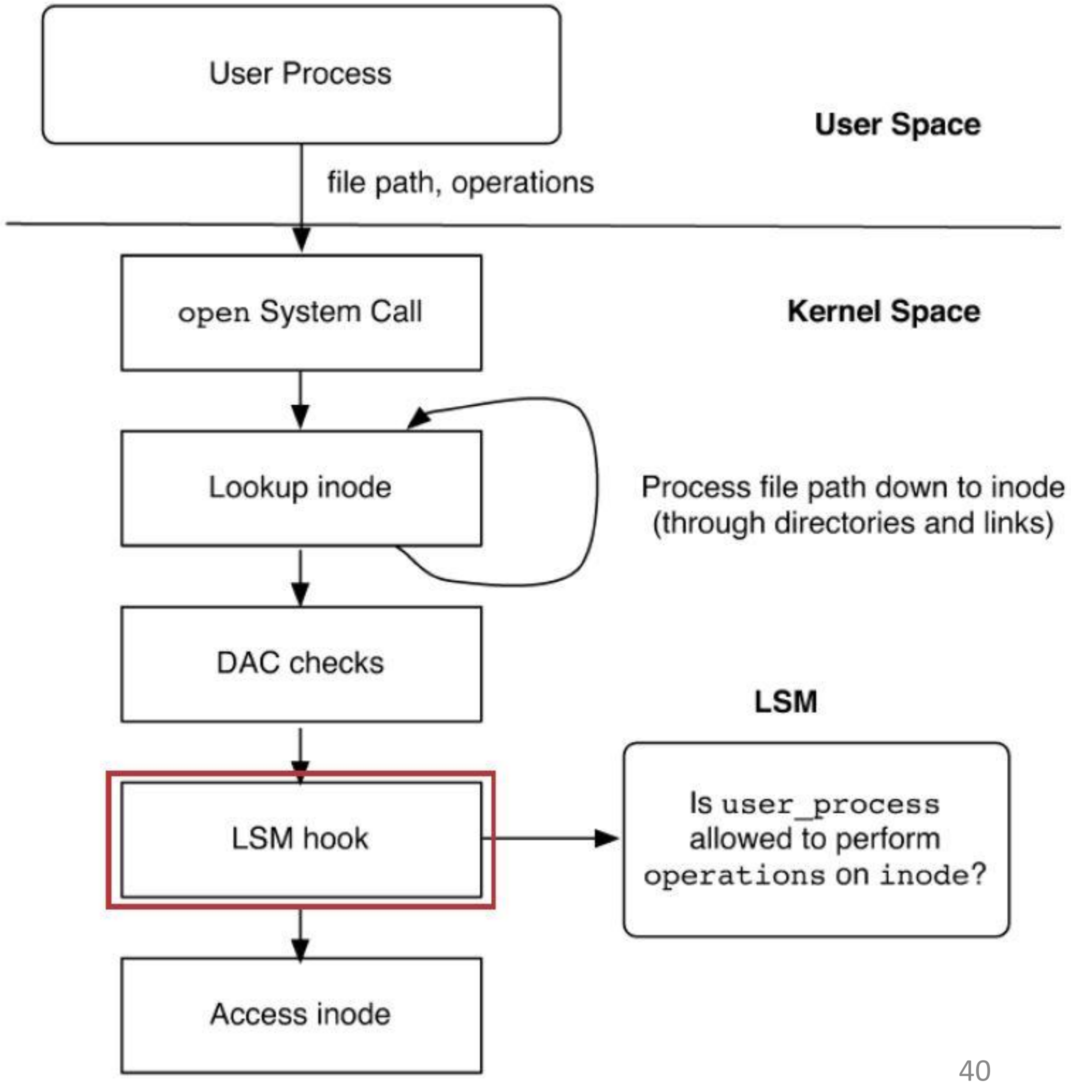
LSM Design – Hooking

- Permissions checks are done by calling LSM hooks.



LSM Design – Hooking

- Open() hook process
  - Process syscall in user
  - Invoke syscall in kernel
  - Lookup inode
  - Check DAC
  - Hook & check MAC
  - Grant access



SELinux History

- Motivated by the security kernel design philosophy
  - But practical considerations were made.
- Developed by the NSA (National Security Agency)
  - Originally published as a kernel patch
  - Integrated in the kernel 2.6
- SELinux has been certified EAL4+ for RHEL 5.
  - In addition to the NSA, RedHat (IBM) actively develops SELinux

SELinux in Linux Distributions

- Fedora: available since kernel 2.
- RedHat: available since RHEL 4.
- Debian: available since 4.0, but deactivated but default.
- Ubuntu: available since 8.04 to replace AppArmor.
- The following was tested in CentOS 7.

SELinux Activation

- In the file */etc/selinux/config*, the option SELinux allows users to choose within three options: *disabled*, *permissive* and *enforcing*.
  - Disabled: not enabled in the kernel or disabled via kernel parameter.
  - Permissive: just logs denials, but does not enforce them.
  - Enforcing: logs and enforces denials for all enforcing domains (processes)

- When SELinux is activated, you can change the mode *permissive* to the mode *enforcing* using the command *setenforce 1*.

- To display SELinux configurations, *sestatus -v*

SELinux at Glance

- Security policies:
  - Centralized store for access control
  - Can be modified by the SELinux system administrator
  - Determined by security contexts (=user, role, type)
  - Specification permissions
  - Labeled with information for each file

- Operations to objects for subjects
  - Append, create, rename, …

SELinux Contexts 1/2

- Processes and files are labeled with an SELinux context that contains additional information:
  - *(ls -Z) ------ root root system_u:object_r:shadow_t:s0 /etc/shadow*
  - The syntax is user:role:type:level

- SELinux User
  - Identity that is authorized for a specific set of roles.
  - Each Linux user is mapped to an SELinux user via the SELinux policy
  - Run semanage login –l to view a list of mappings between SELinux and Linux users.

SELinux Contexts 2/2

- SELinux Role
  - The role is a gateway between a user and a process.
- SELinux Type
  - The type is an attribute of Type Enforcement
  - The type defines a domain for processes and a domain for files
  - Rules define how types can access each other.
- SELinux Level
  - The level is an attribute if MLS (Multi-Level Security) and MCS (Multi-Category Security).
  - Each level is a sensitivity-category pair, with pair being optional.

SELinux User Capabilities

User	Role	Domain	X Window System	su or sudo	Execute in home directory and /tmp/ (default)
sysadm_u	sysadm_r	sysadm_t	yes	su and sudo	yes
staff_u	staff_r	staff_t	yes	only sudo	yes
user_u	user_r	user_t	yes	no	yes
guest_u	guest_r	guest_t	no	no	no
xguest_u	xguest_r	xguest_t	yes	no	no

Domain Transitions 1/2

- A process in one domain transitions to another domain by executing an application that has the *entrypoint* type for the new domain.

Example

- A user wants to change their password. To do so, they run passwd application, that is labeled with the passwd_exec_t type.
  - The passwd accesses /etc/shadow which is labeled with shadow_t type.
- An SELinux policy states that processes running in the passwd_t domain are allowed to read and write files labeled with shadow_t.
  - An SELinux policy states that the passwd_t domain has an entrypoint permission from the passwd_exec_t type.

Domain Transitions 2/2

- When a user runs the passwd application, the user's shell process transitions to the passwd_t domain.
  - And it is done!

- Type Enforcement ensures:
  - The passwd_t domain can only be entered by executing an application labeled with the passwd_exec_t type.
  - Only authorized domains, such as passwd_t, can read and write files with the shadow_t type. Even processes with superuser privileges cannot modify them.
  - Processes in the passwd_t domain can only read and write files labeled with shadow_t type (and the etc_t type).