

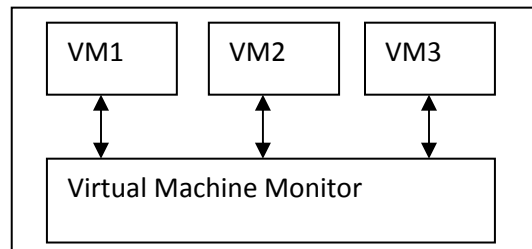
CSE 409/509: Spring 2009, System Security

Access Control: ACMs, HRU Theorem, Bell-Lapadula – 02/06/2009

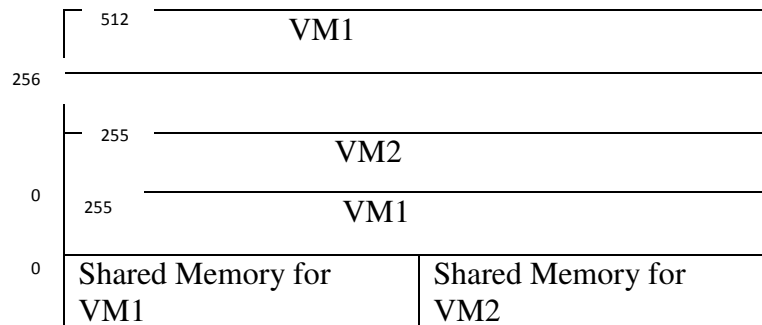
■ Rucha Lale

Sharing of Information:

Virtual machines together with Interrupt handler, TLB, Privileges etc. can be used to implement isolation between programs running on a system. However, this does not promote sharing of information between processes.



One way to achieve sharing of information is to split some part of memory (for e.g a page in memory) between VMs.



For e.g a message passing system that uses a shared page for two VMs to communicate with each other

Possible problems:

1. Synchronization: If two VMs write to a specific memory address what would be the final value at that memory location (can be solved if the VMs agree on which part of memory belongs to whom.)
2. Malicious access: One VM writes to memory belonging to other VM
For e.g VM1 is a File Server and VM2 provides data to be written to a specified file.

Code for VM1

VM2 specifies target file and data

```

target_file = shared_buffer
data = ...
If (permissionsOk(target_file))      ---(I)
    Write(target_file, data)         ---(II)

```

A malicious process on VM2 will first point the target_file to some legitimate or dummy file on the system. If it manages to change the address of the target_file to some other location between (I) and (II) then it can write to a memory location which does not belong to VM2. This introduces the concept of TOCTTOU (Time of Check to Time of Use).

Possible solutions:

1. Use of atomic operations
2. Use of locks
3. Modification of VM1 code

Code for VM1

```

copy_private_buf(private_buf,shared_buf)
target_file = private_buf
data = ...
If (permissionsOk(target_file))      ---(I)
    Write(target_file, data)         ---(II)

```

Here, private_buf being exclusive to VM1, a malicious process on VM2 cannot change the address for target_file once assigned.

Access Control Mechanisms:

1. Access Control Matrix: It consists of
 - a. matrix,
 - b. set of rights and
 - c. set of commands

$$A_{d,x} = \{r \mid d \text{ has } r \text{ access to } x\} \quad (d = \text{domain}, x = \text{object})$$

Domain/Objects	Domain1	Domain2	Domain3	File1	File2
Domain1	Owner			Owner	
Domain2			Control	Read	
Domain3				Write	read

Rights:

- Owner – creator is owner, grant access to objects
- Control – revoke access from domains.
- Copy – useful in case of revocation of rights
- Protected

Commands:

Create , grant, revoke etc.

Access Control Models

1. Harrison Ruzzo Ullman :

Consists of

- a. An Access Control Matrix to start with.
- b. A set of update commands e.g. create/delete objects, create/delete domains, create/delete entries.

General form of commands : $\text{Command}(d_1, r_1, x_1 \dots d_n, r_n, x_n)$

If $(r_1 \in A_{d_1, x_1} \ \& \ r_2 \in A_{d_2, x_1} \ \& \dots)$

{
 Oper1
 Oper2
 ...
}

E.g d_1 wants to delegate read access to d_2 on object x

If $(\text{owner} == A_{d_1, x})$

$A_{d_2, x} = \{r\} \cup A_{d_2, x}$

Question: Given the initial ACM and set of commands can a system ever reach a state satisfying a specific predicate?

Ans: Undecidable. (As the number of objects in the access control matrix can keep on increasing, computation of all possible checks is infeasible.)

2. Bell Lapadula (Multi-level security)

- a. Common in military circle
- b. Deals with confidentiality only
- c. Follows the mandatory access control policy. (Users who create a particular object do not get to decide the read/write permissions over that object.)

Here objects are labeled by their permissions and programs/people/domains which access those objects are labeled by their access permissions.

Following are the possible permissions:

Topsecret	↑ (increasing level of privilege)
Secret	
FOUO (For Official Use Only)	
Public	

Compartments:

A certain area for which a program/person/domain must have clearance to access it.

Eg.g JFK, Area51 etc.

A label for every program/domain/person is of the following format:

Label (L)= (level, set of compartments) where level is the permission level.

Subject = (Ls, Ss)

Object = (Lo,So)

Rules:

1. Subject S can read object O iff $L_s \geq L_o$ & S_s is superset of S_o
e.g if $L_s = (\text{secret}, \{\text{JFK}\})$ and $L_o = (\text{Topsecret}, \{\})$ if subject S wants to read object O, the access is denied.
2. (Write with empty compartments). As the main aim is confidentiality and not integrity, it is ok if a subject S writes to a object O whose level is higher than the subjects.
i.e write is allowed if $L_s \leq L_o$
3. (write with some value for compartments)If the compartments have some values then the subject should have access to a list of compartments that are atleast a subset of the objects compartments.
i.e. write is allowed iff $L_s \leq L_o$ & S_s is a subset of S_o .

So thus, while designing a “copy” program that copies data from one location to other care needs to be taken that the output level has to be at least equal to the input level.