DSCI-519, Assignment 2

Rishit Saiya (rsaiya@usc.edu)

Q1-A:

The Take-Grant model imposes certain conditions on the different types of rights amongst subjects and objects. In Access Control Matrix, there is much more flexibility & it doesn't put restrictions on the types of rights between subject & object. This makes the Take-Grant model relatively rigid as compared to Lampson's Access Control Matrix.

Through the Access Control Matrix, rights can be specified which subject possesses over an object as a function of various parameters like role of the subject, time of a day etc. The Take-Grant model is not capable of being used to check the change in protection state of a system for parameters like role of the subject, time of a day etc. Lampson's access control matrix is much more expressive compared to the Take-Grant Model.

The level of impact on the array of objects can be analyzed using the Access Control Matrix model whereas the case of Take-Grant Model does not include this in the list of allowed permissions/regulations.

Q1-B:

In terms of efficiency, the Take-Grant model does better than Lampson's Access control matrix. The Take-Grant model works at a micro and granular level. Thus, because of its workflow model, the flow of rights across subjects and objects would make it possible to know the amount of trust that system would be placing on different objects. In any event, if the security of the system is compromised with respect to any access rights, the Take-Grant model will allow it to assess all the potential subjects that need to be involved in the exploited right in linear time in general. This aspect is missing in the Access Control Matrix. The Take-Grant model provides more specification & details compared to Access Control Matrix. To a greater extent, it also enables it to analyze whether a theft of that right is possible by a subject without needing the cooperation of any other subject. This characterization of the possible flow of rights across subjects and objects would enable us to quantify the amount of trust that the system would place on different subjects.

Q1-C:

According to the HRU method, the set of commands is said to be safe if no right is leaked.

For substantiating, each property is substantiated as follows:
For Simple Security Policy, let us consider that there are 3 states namely S1 (Top Secret), S2 (Secret), S3 (Confidential). In here, let us consider that the transition/command is T and the access right is "r" (READ). In such a case, if S1 is not reachable from S2 after the execution of T then it implies that "r" access is not leaked with respect to T and S2 state. Hence, the "No Read Up" property is in-tact in such cases.

For *-Security Policy, let us consider that there are 2 states named S1 (Secret) and S2 (Confidential). Here, let us consider that the transition/command is T' and the access right is "w" (READ). In such a case, if S2 is not reachable from S1 after the execution of T then it implies that "w" access is not leaked with respect to T' and S1 state. Hence, the "No Write Up" property is in-tact in such cases.

Q2:

According to the problem statement, the levels given are {Top Secret, Secret}. The categories given are {{Ø}, Europe}.

The BLP MAC Rules applicable to this policy are as follows:
Simple Security: A subject with access class ‘S’ can read objects with access class ‘O’ iff S dom O [No Read Up]
*-Property: A subject with access class ‘S’ can write objects with access class ‘O’ iff O dom S [No Write Down]

[Assumption]:

Objects	**Subjects**
O1: (Top Secret, {Ø}) O2: (Secret, {Ø}) O3: (Top Secret, {Europe}) O4: (Secret, {Europe})	S1: (Top Secret, {Ø}) S2: (Secret, {Ø}) S3: (Top Secret, {Europe}) S4: (Secret, {Europe})

‘r’, ‘w’ are the Read and Write access rights respectively.

Then according to the dom relationships, the Access Control Matrix is as follows:

Subjects/Objects	**O1**	**O2**	**O3**	**O4**
S1	rw	r	w	-
S2	w	rw	w	w
S3	r	r	rw	r
S4	-	r	w	rw

Q3-A:

The notation (k, l, m) are used to define the four ring brackets as asked in the problem statement. The brackets encapsulate the access rights such as ‘r’, ‘w’, ‘e’ (Read, Write and Execute respectively). The relationship between these rings and brackets are as follows:

Access Bracket	**Current Execution Ring**
Read Bracket	R0 - R2
Write Bracket	R0 - R1
Execute Bracket	R1 - R2
Call Bracket	R1 + R2 - R3

In the above current execution ring’s top and bottom of each bracket ranges are defined corresponding to their respective access brackets. If the user’s current ring number is within the allowed ring bracket, then the user is given the appropriate access. It is typically the case where the access rights are given to the user according to the current ring placement in the range.

In theory, the access brackets only contain one ring. But for the purpose of optimisation, the operating system latency/overhead caused by ring crosses which would occur each time a procedure attempts to use a function access file finally culminates to procedures being placed within the routine’s access bracket in the ring heirarchy.
The call bracket here is referred to as the entity which allows controlled changes to the privilege and eventually access which makes it feasible for the operations like kernel operations on Operating System to execute on behalf

(proxy) of users, whilst also mitigating the exploitation of temporary privilege (explained next). If a segment which the user wishes to execute is not within the range of the user's execute bracket/call bracket, then the user is granted access temporarily and the current ring number is changed accordingly to the highest number in the segment's execute bracket which the user is using.

The gates are defined as the segments with non-null call brackets. It is so because user processes can only pass from outer to inner ring segments through them. In common, the simply stated call brackets often use gates as their default entry point between the rings to provide users with controlled access rights to invoke a routine not within their execute bracket but still within the call bracket of the user.

Q3-B:

[Assumption]: The access rights of read, write, execute are denoted here as - {r, w, x} respectively.

1. k = 4, l = 4, m = 6; read flag: on, write flag: on, execute flag: on; process current ring = 4
Access of r, w, x | Reason: r = k
2. k = 0, l = 4, m = 6; read flag: on, write flag: on, execute flag: on; process current ring = 7
All Access Denied | Reason: m < r
3. k = 4, l = 4, m = 6; read flag: on, write flag: on, execute flag: on; process current ring = 3
Access r, w, x | Reason: r < k and hence for e (execute) ring fault occurs; setting r = k & hence given access of r, w, x
4. k = 0, l = 4, m = 6; read flag: on, write flag: on, execute flag: off; process current ring = 6
All Access Denied | Reason: l < r ≤ m, also however execute flag is off
5. k = 0, l = 4, m = 6; read flag: on, write flag: on, execute flag: on; process current ring = 5
Execute | Reason: l < r ≤ m, e permitted through gate, transition to more privileged ring

Q4:

[Assumption]: All the derivations towards the users, their corresponding clearances and objects, their corresponding classes are being referenced from the reference book: Computer Security Art and Science [Bishop] (Section 6.3.1)

The label assignment according to which these requirements are satisfied as follows in the similar order:
1. The subjects ordinary users and the object of Production Code have the same security level of (SL, {SP}). This is in accordance with the simple security and *-property of BLP model which essentially grants the aforementioned subjects the access rights of read and write to the object. By assigning Production Code (IO, {IP}) at the Integrity Level, an ordinary user cannot alter the Production Data. This is in accordance with the Biba model.
2. Production Data can be assigned in (ISL, {IP}) Integrity Class and hence an ordinary user can alter and read the Production Data.
3. The system managers need access to all logs but cannot change levels of objects due the assignments of labels which are (SL, {SD, T}). This requirements satisfies as the conditions which are satisfied here are as follows:
(ISL, {IP, ID}) dom (ISL, {IP}) AND (SL, {SP}) dom (SL, {Ø}) for establishing domination relationships and showing they can access all logs.
4. System Controllers cannot access the development code/test data security categories, system programs in modification since the integrity, or software tools since the integrity policy.
5. The system and applications logs are to have only append-only access, which is satisfied by the following conditions:
(ISL, {IP}) dom (ISL, {Ø}) AND (AM, {SP}) dom (SL, {SP}) for establishing domination relationships and showing they can only access the append-only right. The two security levels defined here are {AM, SL} from higher to lower. (AM: Audit Manager, SL: System Low). The five categories are defined as ({D, PC, PD, SD, T}) which are Development, Production Code, Production Data, System Development, Software Tools respectively.

The assignment of labels follows the notions as follows:

- Ordinary users use production code to modify production data and hence, their clearance is (SL, {PC, PD}).
- The application developers in the organization need access to tools for developing their programs. In order to facilitate that and to a category for the programs that are being developed (the categories should be separate). Hence, application programmers have (SL, {D, T}) clearance. System programmers develop system programs and, like application programmers, use tools to do so; hence, system programmers should have clearance (SL, {SD, T}).
- System managers and auditors need system high clearance, because they must be able to access all logs and hence their clearance is (AM, {D, PC, PD, SD, T}).
- The system controllers must have the ability to downgrade the level of code access once it is certified for production. This is done so that other entities cannot write to it. Hence, the clearance for this type of user is (SL, {D, PC, PD, SD, T}) with the ability to downgrade programs.
- Objects that might be altered have two categories: that of the data itself and that of the program that may alter it. For example, an ordinary user needs to execute production code; hence, that user must be able to read production code. Placing production code in the level (SL, {PC}) allows such access by the simple security property of the Bell-LaPadula Model. Because an ordinary user needs to alter production data, the *-property dictates that production data be in (SL, {PC, PD}) as labels.

Users	**Clearance**
Ordinary Users	(SL, {PC, PD})
Application Developers	(SL, {D, T})
System Programmers	(SL, {SD, T})
System Managers and Auditors	(AM, {D, PC, PD, SD, T})
System Controllers	(SL, {D, PC, PD, SD, T}) and Downgrade Privileges

The objects that might be altered have two categories in general - Data and the Program that altered it and they are assigned certain security levels based on who should access them and who should not.

1. Due to the users not having execute access to category T, they cannot write their own programs, hence Requirement 1 is met.
2. The application programmers and system programmers do not have access rights to read or write to category PD, and therefore cannot access the production data. If they do require production data to test their programs then the data must be downgraded from level PD to D. It cannot be upgraded because the model has no upgrade privilege in place. The downgrading of the level requires an intervention of system control users which is a special process within the meaning of Requirement 2. Hence, Requirement 2 is satisfied.
3. During the process of installing a program, it requires the downgrade privilege specifically, while changing the category of the program from D to PC level. This belongs only to the system control users and hence, only those users can install applications or system programs. The need for a special process for use of the downgrade privilege satisfies Requirement 3.
4. By allowing only system control users to have the downgrade privilege, the auditing part is met by requiring all downgrading to be logged and hence the control part of the Requirement 4 is met.
5. The placement of system management and audit users in Audit Manager level ensures that they have access to both - system state and system logs. Hence the model meets the Requirement 5 also.

In this way, these requirements are satisfied by the label assignment by showing the effects of applying BLP and Biba rules to the labels of relevant object and subject types.