CSC648 Homework 2

Shuowen Wei

1. Since confidentiality is the objective, thus we should follow the rules of Bell-LaPadula, which concerns secrecy by “no read up” and “no write down” to protect information from leaking.

A.

Since {A, C} is at a higher level than {B, C}, thus {A, C} can only read {B, C}, but can’t write to {B, C}.

B.

Since {C} and {B} are at the same level, thus {C} can not only read but also write to {B}.

C.

Since Nairb is at a higher level than (confidential, {C}), thus Nairb can only read (confidential, {C}), but can’t write.

2.

I would suggest Role-Based Access Control (RBAC) mechanism for this company. First of all, since there might be a lot of objects (three versions of the codes) and subject (four types of employees) in this company, it is really inconvenient and inefficient to maintain a large dimension of ACM, so it is the same with long ACLs and C-lists since ACL is just a column of ACM and C-list is just a row of ACM. Secondly, considering each type of the employee in this company basically deals with different object, there duties are mutually separate, and this is exactly the model that RBAC works on, because RBAC can prevent subjects in certain roles from entering other roles.

3.

Well, as mentioned in question 1, Bell-LaPadula model prevents “read up” and “write down” for protecting information from being leak. So, if these two active subjects that wish to send and receive signals to each other, they can only be at the same level, or the subject at a higher level can’t send signals out and the one at a lower level can’t receive signals.

4.

Since the default of SELinux uses combination of RBAC and Multi-Level Security (MLS), the labels of subjects (users) in SELinux should contain the generic identity for the OS to indentify, the role values used for the files, and type of this subject which determines how the file can be accessed (read/write/execute/copy/delete/e.t.c).

I guess, in each role level, there are multiple security sub-levels …

5.

Android application’s framework requires programmers to use a certain structure, and there is a launchpoint is specified in the manifest which every package should include. The OS in Android uses Dalvik virtual machine, the main entry point to the execution of the application is encapsulated within the framework. The OS knows the beginning of an application when the first call to [*onCreate()*](http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) happens, which initialize your activities. Programs don’t have *main()* allows for sharing of code between applications.

6.

New relation: System must maintain certified relations and ensure only user’s certified to run on a CDI change that CDI, and associate such certified user a set of CDIs.

The reason why the original model didn’t do this might be that, considering users may changes frequently while transformation procedures may remain the consistent for a longer time, it is more convenient just to associate a user with a TP rather than merely a set of CDIs (bind the certified TP with its set of CDIs together).