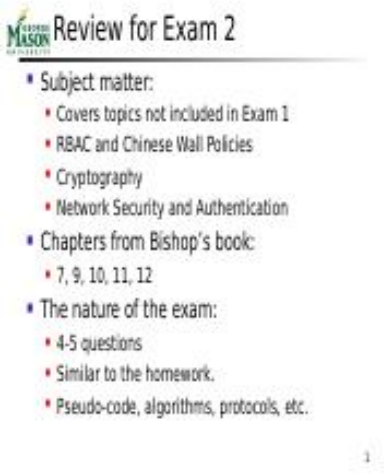
HW2.docx - ISA 562 Assignment 2 1 Exercise 3.9#1 The proof...

SCHOOL	COURSE TITLE	UPLOADED BY	PAGES	RATINGS
George Mason ...	ISA 562	xincro	7	100% (1) 1 out of 1 people found this document helpful

ISA 562 – Assignment 2

1. **Exercise 3.9, #1.**
   The proof of Theorem 3-1 states the following: Suppose two subjects s1 and s2 a created and the rights in A[s1,o1] and A[s2,o2] are tested. The same test for A[s1,o and A[s1,o2] = A[s1,o2] U A[s2,o2] will produce the same result. Justify th statement. Would it be true if one could test for the absence of rights as well as fo the presence of rights?

   Ans.



are checked for presence of rights.

Now considering the same test for say A[s1,o1] and A[s1,o2] (i.e,. A[s1,o1] A[s2,o2]) the check presents the same output as earlier as A[s1,o1] is same whereas cas of A[s1,o2] the matrix A[s2,o2] in the earlier case is added some rights to get A[s1,o2 So, hence the check as usual checks for presence of rights which are present in A[s1,o as in A[s2,o2] hence produces the same result.

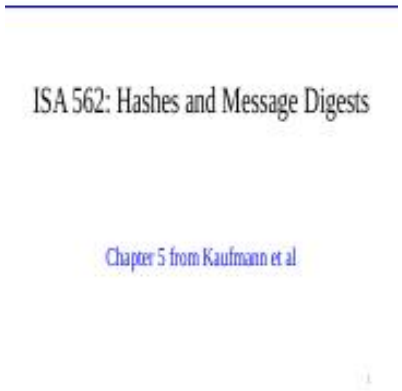
Now if we check for absence pf rights. Say we are checking for a particul absence of rights ‘a’ then on checking for it in A[s1,o1] and A[s2,o2], it might succee but it will not be the same result as checking A[s1,o2] as the absence of rights ‘a’ migl be added to it and hence might fail

2. **Exercise 4.11, #1.**
In Figure 4-1, suppose that edge t3 went from s1 to s4. Would the resulting syste be secure?

Ans. In the given system s1 and s2 are the only authorized states while s3 and s4 a unauthorized states. A = { s1 , s2 } ; UA = { s3 , s4 }

The edge t3 actually goes from s1 to s3 and is not secure. Suppose it goes from s1 to s the system will still not be secure because s4 is an unauthorized state just like s3. Goi from s1 to s4 means going from authorized to unauthorized states, so the system will n be secure.

3. **Exercise 4.11, #5.**
Classify each of the following as an example of a mandatory, discretionary, originator controlled policy, or a combination thereof. Justify your answers.



a. The file access control mechanisms of the UNIX operating system.

Ans. It is Mandatory Access Control since the system controls access to an object ar permissions to users can also be controlled by system. Typically, the system mechanis will check information associated with both the subject and the object to determi whether the subject should access the object. It can also be Discretionary Access Contr since users can give access to an object.

b. A system in which no memorandum can be distributed without the author consent.

Ans. It is Originated Controlled Access Control since no memorandum can be distribute without the author's consent. Actually the owner of the file has no control over who may access, the main goal of this control is to allow the originator (author) of the file to control the dissemination of information

c. A military facility in which only generals can enter a particular room.

Ans. It is a mandatory controlled policy. Here the assumption is that a particular room is secured by system mechanism which allows access only to generals. The individuals cannot alter this access. But, if we assume that the particular room is controlled by an individual who also has the power to designate people as 'generals', then this would become a discretionary access control.

d. A university registrar's office, in which a faculty member can see the grades of a particular student provided that the student has given written permission for the faculty member to see them.

Ans. Discretionary Access Control because an individual student's permission is needed for a faculty member to see the grades of a particular student. It can also be Originator Controlled Access Control because the originator (i.e University Registrar) can control a student's record.

4. Exercise 4.13, #6.
A process may send a message to another process provided that the recipient is willing to accept messages. The following class and methods are relevant.

class Messages {

Ans. deny () = Messages.deposit() when (Messages.withdraw() = = 0);

5. **Exercise 4.1(i), #7.**
   Use BSD4.4, to create a domain *d_guest* composed of processes executing the restricted shell /usr/bin/rsash. These processes cannot create any files. They can read and execute any object of type t_public. They can read and search any object of type t_guest.

   Ans. type t_symlink t_guest
   domain d_guest = {/usr/bin/rsash};
   (rx, t_public);
   (rl0, t_guest);

6. **Fred Schneider's book, Chapter 7, Ex 7.2**
   In some systems, the same object might have different names. We see this in many guises: (i) when shared memory segments are mapped to different virtual addresses in different processes, (ii) when aliasing occurs in a high-level language program through parameter passing or the use of pointers, (iii) when either a symbolic or a hard "link" is created for accessing the same file from different directories, and (iv) when a device in a network is named by its networks address and that device is "multi-homed".
   a. **What, if any, opportunities could this functionality provide to attackers if O in (P, O, op) could be one of many names that principals use for the object.**

   Ans. If the object O is identifying the same unique object each time irrespective of what principal (in this case, the principal is end up is one of the privilege among the set of privileges P(o)) is exercising, then the transferring capabilities or even moving the names might have accidental consequences.

   For example, In this case the object name O designates to some object Obj but is later recycled to designate to some other object Obj', then by holding long enough, a principal eventually gets privileges to access Obj' (even though access to Obj was what had been authorized). And if an object name O designates object Obj in one principal's address space but Obj' in another's, then transferring unintentionally gives the recipient (attacker) privileges to Obj' (even though a capability for Obj was being transferred).

TERM	PROFESSOR
Fall '08	**Staff**