Homework #2

Problem 1 (10 pts) Explain the following terminologies: Trusted Computer Base, discretionary access control, mandatory access control, covert channels. Also, what are the differences between the concept of "trusted" and "trustworthy"?

Answer.

Trusted Computer Base Set of hardware and software that is important to the system security policy. The security of system depends on the correct behaviors of the trusted computing base.

Discretionary Access Control is a type of access control that restricts access to objects based on the identity of subjects and groups to which they belong. One property of DAC is that a subject can pass its access rights to any other subjects.

Mandatory Access Control is a type of access control that restricts the access of subjects to objects based on a system-wide policy. When a subject performs an access to an object, the system will use a predefined policy (normally set by system administrator) to verify the access right of the subject on the subject.

Covert Channel is a mechanism to transfer information between 2 parties without revealing any traces of communication to the system/network administrator and other observers. Depends on the context, Covert channel is a transmission channel that is used to transfer data in a manner that violate security policy, or it also can be seen as a tool to bypass censorship authority.

"Trusted" vs "Trustworthy" a "trusted" component of a system is determined by its role in the system, and it means that the security of the system depends on that component, a failure of the trusted component can break the security policy, and . A "trustworthy" component of a system is determined by intrinsic properties of the component, and it means that the component deserves to be trusted (e.g implemented correctly).

Problem 2 (20 pts) The description of this problem is long, and the question you need to answer are in boldface.

In the Bell-LaPadula (BLP) model, a computer system is modeled as a state-transition system. To use the system model in BLP to describe a computer system, one first specifies S, S_T, A, L, which are explained as follows.

- S is a set of subjects.

  The set of subjects in a system is fixed in the original description of BLP. This means that when a system is running, no new subject can be added and no subject can be removed. This certainly limits the kinds of systems that can be modeled in the BLP system model. It is possible to extend the BLP system model to allow a dynamic set of subjects.

- A set $S_T \subset S$ of trusted subjects.
- A set of access modes A. For simplicity, we assume that $A = \{read, write\}$.
- A partially order set $\langle L, \leq \rangle$ of security levels.

Each state is a tuple $\langle O, b, F \rangle$:

- O is the set of the objects that currently exist in the state.
- $b \subseteq S \times O \times A$ is the current access set. Each element in b is a triple (s, o, a), meaning that currently the subject s holds the access mode a over the object o.
- $F = \langle f_S, f_O, f_C \rangle$ is a triple of security level functions, where
  - $f_S : S \longrightarrow L$ is the (maximum) security level function for subjects,
  - $f_O : O \longrightarrow L$ is the object security level function, and
  - $f_C : S \longrightarrow L$ is the current security level function for subjects.
    The idea is that while a subject does not always have to operate at its maximum security level. A subject can change its current security level.

A system is BLP-secure if and only if every reachable state is a secure state. A state $\langle O, b, F = \langle f_S, f_O, f_C \rangle \rangle$ is *a secure state* if and only if it has the following two properties:

1. it satisfies *the simple-security property (ss-property)*; that is, for every $(s, o, read) \in b$, $f_S(s) \geq f_O(o)$. In other words, if a subject can read an object, the subject's *maximal* security level must dominate the object's security level.
2. it satisfies *the *-property*; that is, for every $(s, o, a) \in b$ where $s \notin S_T$, the following conditions hold
   - if $a = read$, then $f_C(s) \geq f_O(o)$
   - if $a = write$, then $f_C(s) \leq f_O(o)$

   Note that the current security level of a subject is used in the *-property.

- **Several authors said that the *-property implies the ss-property. Why is this wrong? Under what conditions would the *-property imply the ss-property?**

  Answer: **The *-property applies only to those subjects are not trusted subjects. In other word, *-property doesn't affect trusted object. For subject $s \in S_T$, s is still able to read o despite $f_S(s) < f_O(o)$, and s is able to write o despite $f_S(s) > f_O(o)$.**
  If $S_T = \emptyset$, the *-property implies ss-property. Also, If we modify that *-property should apply to all subjects not just trusted subject, then *-property will imply ss-property.

- **The definition of the *-property above has two conditions, one for reading and one for writing. Can we remove the restriction about reading? Explain why.**

  Answer **Yes. But we need to apply the tranquility principle: subject cannot change current levels or cannot drop to below the highest level read so far. The reason is that, if we remove the read property of *-property, the ss-property still apply on the subject. The subject can use its maximum security level to read an object, and since we restrict**

that subject cannot drop its level to lower level, it should not able to leak information about the object to to a lower level.

Consider the following system, with $S = \{s_1, s_2\}$, $S_T = \emptyset$, $L = \{\mathsf{high}, \mathsf{low}\}$. The following are the only state transitions that are allowed:

- A subject s request to read an object o, which succeeds only if adding $(s, o, read)$ to b does not violate the ss-property and the *-property.
- A subject s stops reading an object o, which succeeds if $(s, o, read)$ is in b, and results in $(s, o, read)$ being removed from b.
- A subject s request to write an object o, which succeeds only if adding $(s, o, write)$ to b does not violate the ss-property and the *-property.
- A subject s stops writing an object o, which succeeds if $(s, o, write)$ is in b, and results in $(s, o, write)$ being removed from b.
- Change the current security level of a subject, which succeeds if after the change, the ss-property and *-property are satisfied.

The initial state is specified as follows:

- $O = \{o_1, o_2\}$,
- $b = \{(s_1, o_1, read)\}$,
- $f_S = \{s_1 \rightarrow \mathsf{high}, s_2 \rightarrow \mathsf{low}\}$,
- $f_O = \{o_1 \rightarrow \mathsf{high}, o_2 \rightarrow \mathsf{low}\}$,
- $f_C = \{s_1 \rightarrow \mathsf{high}, s_2 \rightarrow \mathsf{low}\}\}$.

- **Prove, using mathematical induction, that the system is BLP-secure.**

Answer. **Let the initial state $S_0 = \{O_0, b_0, (f_S, f_O, f_C)_0\}$ defined as above.**
Base case: **The initial state S_0 satisfies ss-property because $f_S(s_1) = f_O(o) = \mathsf{high}$. It also satisfies *-property because $f_C(s_1) = f_O(o) = \mathsf{high}$**
Induction Step: **Let assume that $S_k = \{O_k, b_k, (f_S, f_O, f_C)_k\}$ is a secure state. We want to prove that S_{k+1} is also secure state. Since there are only 5 state transitions are allowed, we consider each transition:**

- **A subject s request to read an object o, which succeeds only if adding $(s, o, read)$ to b does not violate the ss-property and the *-property. In other word, since every current access set in $b_k \in S_k$ does not violate ss-property and *-property, every accesses in $b_{k+1} \in S_{k+1}$ do not violate ss- and *-property. Therefore, S_{k+1} is secure state.**
- **A subject s stops reading an object o, which succeeds if $(s, o, read)$ is in b, and results in $(s, o, read)$ being removed from b. Similar, removing one some accesses from current access set should not violate ss- and *-property. Hence, S_{k+1} is secure.**
- **A subject s request to write an object o, which succeeds only if adding $(s, o, write)$ to b does not violate the ss-property and the *-property. This is similar to first**

transition; therefore, using same argument, S_{k+1} is secure

- **A subject s stops writing an object o, which succeeds if $(s, o, write)$ is in b, and results in $(s, o, write)$ being removed from b. This is similar to second transition; therefore, using same argument, S_{k+1} is secure**
- **Change the current security level of a subject, which succeeds if after the change, the ss-property and *-property are satisfied. Again, this implies S_{k+1} is secure state.**

We showed that for all possible transitions from S_k to S_{k+1}, if S_k is secure state, then S_{k+1} is secure state. We complete our proof.

- **Provide a sequence of state transitions such that in the sequence, the subject s_1 reads o_1, writes to o_2, which is then read by s_2. That is, we show that the system is intuitively insecure, even though it can proven secure under the BLP definition.**

Answer.

1. **s_1 requests to read o_1. This doesn't violate ss- and *-properties, so now $b = \{(s_1, o_1, read)\}$.**
2. **s_1 stops reading o_1. This doesn't violate ss- and *-properties, so now $b = \{\}$**
3. **s_1 changes its current security level to low. So $f_C = \{s_1 \rightarrow \mathsf{low}, s_2 \rightarrow \mathsf{low}\}$**
4. **s_1 requests to write o_2. We have $b = \{(s_1, o_2, write)\}$. At this point, s_1 can write content of o_1 to o_2**
5. **s_1 stops writing o_2. We have $b = \{\}$**
6. **s_2 requests to read o_2. We have $b = \{(s_2, o_2, read)\}$. At this point, s_2 can read content of o_1 in o_2**

Thus, we showed that s_1 can leak information by dropping its security level, and it breaks the confidentiality despite BLP is proven to be secure.

- **How to fix the BLP security definition so that system with the above weaknesses will not be proven secure?**

Answer. **We should apply the tranquility principle in which subject cannot change current levels or cannot drop to below the highest level read so far.**

Question 3 (35 pts) Read the following papers, and explore the Biba integrity model.

- N. Li, Z. Mao, H. Chen: "Usable Mandatory Integrity Protection for Operating Systems". In IEEE Symposium on Security and Privacy, May 2007. http://ieeexplore.ieee.org/document/4223222/
- Ziqing Mao, Ninghui Li, Hong Chen, Xuxian Jiang: "Combining Discretionary Policy with Mandatory Information Flow in Operating Systems." ACM Trans. Inf. Syst. Secur. 14(3): 24:1-24:27(2011) https://dl.acm.org/citation.cfm?id=2043624

Answer the following questions.

- **Identify two attack scenarios where the attacks are prevented by UMIP and/or IFEDAC. Be specific about your assumptions.**

Answer.

- **UMIP and IFEDAC provide protection to network-based attacks. For example, An adversary can try to remotely install a rootkit remotely into the system by having user to activate malicious program. Such attacker will not succeed because in UMIP, the process will be dropped to low-integrity integrity because it tries to execute remote network traffic, and a low integrity process doesn't have write access on sensitive data. by using integrity tracking, IFEDAC denotes remote traffic file as `net`; therefore, any process that execute that file is tagged with net.**
- **IFEDAC prevents malicious user (or user with weak password) from compromising system protecting by maintaining integrity level for each process. Under some reasonable assumptions, all allowed operations reflect the intention of authorized users.**

- **Identify two attack scenarios where the attacks can bypass UMIP and/or IFEDAC. Be specific about your assumptions.**

Answer.

- **In UMIP, a malicious user who reside on local system can launch an attack to the system. The reason is that user files is normally treat as high integrity; therefore, a malicious/compromised user can modify the file and execute it. The process that execute the file will be dropped to a low-integrity process. Thus, UMIP cannot prevent such attack.**
- **In both IFEDAC and UMIP, if a legitimate user/root user are compromised, and those users decide to upgrade a malicious file's integrity level. Then there is nothing we can do about it. Moreover, in IFEDAC, there are some assumptions for the system such as hardware is not compromised, kernel cannot be exploited... Thus, if one of those assumption is wrong, then there will be an attack that by pass IFEDAC.**

Observe that in integrity protection, when a low-level subject attempts to write to a high-level object, there are three choices: (W1) Forbid it; (W2) Allow it, but drops the integrity level of object after the writing; (W3) Allow it, without changing the object's level.

Also, when a high-level subject attempts to read a low-level object, there are three choices; (R1) Forbid it; (R2) Allow it, but drops the integrity Level of subject after reading; (R3) Allow it, without dropping the subject's integrity level.

- **Describe the five integrity policies in the Biba model by identifying which of the three choices for reading and three choices for writing are used in each of the five models.**

Answer.

1. **Strict Integrity Policy: (W1) (R1)**
2. **Subject Low-Water Mark Policy: (W1) (R2)**
3. **Object Low-Water Mark Policy: (W2) (R2)**
4. **Low-water mark integrity audit policy: (R2) (W2)**

5. **Ring policy: (W1) (R3)**

- **For each of the six choices, identify (a) which kind of trust (if any) it places on the subject; (b) whether the object's integrity level indicate quality or importance; (c) whether it is used in UMIP and/or IFEDAC?**

Answer.

	Kind of trust	object's integrity level	UMIP or IFEDAC
W1	No Trust	Importance	both UMIP and IFEDAC
W2	No Trust	Quality	UMIP
W3	Trusted	Importance	None
R1	No Trust	Importance	IFEDAC
R2	No Trust	Importance	both UMIP and IFEDAC
R3	Trusted	Importance	both UMIP and IFEDAC

- **What additional differences UMIP has when compared with the Biba model, beyond the choices in the R/W rules?**

Answer.

  - **UMIP supports a number of ways to specify some programs as partially trusted to allow them to violate the default contamination rule or the default restrictions on low integrity processes in some limited way. In particular, low integrity objects can be upgraded to hight by a high integrity object, and low integrity objects can access high protected objects via exception.**
  - **In UMIP a file essentially has two integrity level values: whether it is protected and whether it is contaminated**
  - **UMIPs integrity protection is compartmentalized by users. Even if one user has an exception policy that allows all low-integrity processes to access certain files owned by the user, another users low-integrity process is forbidden from such access.**
  - **UMIP allows low-integrity files to be upgraded to high-integrity. (This feature also exists in LOMAC.) This means that low-integrity information (such as files downloaded from the Internet) can flow into high-integrity objects (such as system binaries); however, such upgrade must occur explicitly, i.e., by invoking a special program in a high-integrity channel to remove the sticky bit. Allowing such channels is necessary for patching and system ungrade**
  - **UMIP offers confidentiality protection. In particular, low integrity process doesn't allow to read files owned by system account. Biba only offer integrity protection.**
  - **UMIP uses DAC information to determine integrity and confidentiality labels for objects, whereas in LO- MAC each installation requires manual specification of a mapping between existing files and integrity levels.**

- **What are the main differences between UMIP and IFEDAC?**

Answer.

- **UMIP only have two integrity levels, and IFEDAC has multiple integrity levels combined with read/write protections classes.**
- **while both model provides protection against network-based attacks, IFEDAC provides protection against local attackers which is a weakness of UMIP.**
- **UMIP was designed to make current DAC mechanism more usable, while IFEDAC tries to fix the existing DAC's weaknesses.**

Problem 4 Clark-Wilson (15 points) Read the Clark-Wilson paper.

- D.D. Clark and D.R. Wilson. "A Comparison of Commercial and Military Computer Security Policies".

Answer the following questions.

- **How would you compare the Biba integrity models and the Clark-Wilson integrity model?**

Answer.

- **Biba model lacks the procedures and requirements to identify subjects as trusted. In particular, in Biba model, the conversion from unconstrained data item (e.g. data input) to constrained data item can be done only by a security officer or trusted process; however, in Clark-Wilson model, this restriction is unrealistic. They argued that data input is the most common system function, and it should not be done by a mechanism outside the security model.**
- **Clark-Wilson model focuses on how to verify and ensure that programs can be trusted. In order to do that, they introduces the integrity policy that is defined by two classes of procedures: Integrity Verification Procedures (IVPs) and Transformation Procedures(TPs). IVPs corresponds to audit procedure, and TPs corresponds to the concept of well-formed transactions.**

- **List the two or three most significant new insights you took away from the Clark-Wilson paper and the most significant flaws or weaknesses of it (if any).**

Answer. **Significant insights:**

- **Clark-Wilson model focuses on two mechanisms of fraud and error control which are well-formed transaction and separation of duties. The concept of the well-formed transaction is that a user should not manipulate data arbitrarily, but only in constrained way; in other word, Data can be only manipulated through trusted code. Separation of duties is to separation all operations into several subparts and require each subpart be executed by different programs.**
- **The paper proposed a commercial evaluation criteria that is very useful for integrity control. First, the system must separately authenticate and identify every user, so that his actions can be controlled and audited. Second, the system should**

control the access to data; in particular, a data item can be manipulated only by specific set of programs. Third, the system must associate with each user a valid set of programs to be run, and the data center controls must ensure that these sets meet the separation of duty rule. Fourth, the system must maintain an audit log that records every program executed and the authorizing user. Moreover, the computer system must contain mechanisms to ensure that the system enforces its requirement, and the mechanism must be protected against unauthorized change.

Weakness:

- **The separation of duties mechanism may not work if there is a collusion between employees or compromised programs. the "no collusion" assumption may not be suitable for computer system where multiple programs can be compromised.**

Problem 5 (20 pts) Read the following paper.

- R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-Based Access Control Models. IEEE Computer, 29(2):38–47, February 1996.

And answer the questions below:

- **Why is the notion of roles a useful concept? What are the differences between roles and groups?**

  Answer. **The notion of roles is useful concept because:**

  - **Using roles reduce the number of relationships to manage. It's possible to reduce the number of relationships from $\mathcal{O}(mn)$ to $\mathcal{O}(m+n)$**
  - **Roles add an useful level of abstraction**
  - **Suitable for organizations, as they operate based on roles**
  - **Role is more stable than collection of users, and permissions associated with it.**

  Differences between roles and groups:

  - **Groups are typically treated as a collection of users and not as collections of permission. a role is both a collection of users on one side and a collection of permissions on the other side.**
  - **Role serves as an intermediary to bring two collections of users and permissions together.**
  - **It is easier to determine membership of a group than to determine the permissions of the group. For a role, it should be approximately easy to determine role membership and to determine role permission.**
  - **Roles can be activated and deactivated, and group cannot**

- **Briefly describe the four models in the paper: RBAC_0, RBAC_1, RBAC_2, and RBAC_3.**

  Answer.
  RBAC_0: is the based model. It indicates the minimum requirement for any system that professes to support RBAC. RBAC_0 has following components:

  - **U, R, P, S (users, roles, permissions, and sessions)**

- **Static relations:** $PA \subseteq P \times R$ **a many-to-many permission to role relation.** $UA \subseteq U \times R$**, a many-to-many user to role relation.**
- **Dynamic relations:** $user : S \rightarrow U$ **function mapping session** s_i **to single user** $user(s_i)$**.** $roles : S \rightarrow 2^R$ **a function mapping each session** s_i **to a set of roles** $roles(s_i) \subseteq \{r|(user(s_i), r) \in UA\}$
  The session s_i **has permission** $\cup_{r \in roles(s_i)}\{p|(p, r) \in PA\}$

RBAC$_1$: introduce role hierarchies. Role hierarchies are means for structuring roles to reflect an organization's lines of authority and responsibility. Moreover, role hierarchies implies user inheritance, permission inheritance, and activation inheritance. In RBAC$_1$, these hierarchies are modeled as partial orders relations. RBAC$_1$ has:

- U, R, P, S, PA, UA**, and user unchanged from RBAC$_0$**
- $RH \subseteq R \times R$ **is partial order on** R **called the role hierarchy, Also written as** $\geq$**.**
- $roles : S \rightarrow 2^R$ **is modified from RBAC$_0$ to require** $roles(s_u) \subseteq \{r|(\exists r' \geq r)[(users(s_i), r') \in UA]\}$ **and session** s_i **has the permission** $\cup_{r \in roles(s_i)}\{p|(\exists r'' \leq r)[(p, r'') \in PA]\}$

RBAC$_2$: is unchanged from RBAC$_0$ except there are a set of constraints that determine whether or not values of various component of RBAC$_0$ are acceptable. Only acceptable values are permitted. Constraints correspond to higher-level organizational policy. One example of constraint is the static separation of duty constraint which restricts that no user can hold both roles; this constraint prevents users from having too much permissions. Dynamic separation of duty constraint restricts that no users can activate both roles in one session Another example of constraint is cardinality constraint in which it restricts the number of users for a role or the number of users can activate a role in one session.
RBAC$_3$: is the consolidated model that combined both RBAC$_1$ and RBAC$_2$ to provide both hierarchies and constraints.

- **Describe the constraints considered in the paper, and for each type of constraints discuss whether they are related to the "Separation of privilege" and "least privilege" principles identified by Salzer and Schroeder.**

Answer. **Constraints considered in the paper are:**

- **Mutually exclusive roles: This constraint is related to the "separation of privilege". One example of this constraint is to restrict the same user to be assigned to at most one role in a mutually exclusive set. This constraint provides additional assurance for separation of duties.**
- **Cardinality constraints: This constraint is related to the "least privilege". This constraint control the distribution of powerful permission in a system.**
- **Prerequisite Roles: This constraint is related to the "separation of privilege". It requires user to meet certain conditions to be member of some role. Thus, user are separated based on competency and appropriateness.**

- **People often claim that RBAC is natural to support policies such as separation of duty and least privilege. Give your thoughts on such claims. In particular, how can one justify such claims? How can one criticize such claims?**

Answer. **The claim is not true. The reason is that it really depends on which RBAC model and policy are used. It's reasonable to assume that $RBAC_2$ and $RBAC_3$ support separation of privilege and least privilege because those two models use the concept of constraint to separate permissions between users, but this really depends on the constraints set by system administrator. However, this also implies that the separation of privilege and least privilege mechanisms are not the goals of the RBAC models . Also, $RBAC_0$ and $RBAC_1$ do not support such mechanisms. In particular, $RBAC_1$ with its inheritance properties actually violate the least privilege and separation of privilege principle. An**