Course Logistics

- Homework 4 assigned Due Monday.
- Homework 5 assigned.
- VPN running and available.
Security Policy

- A security policy is a set of rules stating which actions or permitted and which are not.
- Can be informal or highly mathematical.
- If we consider a computer system to be a finite state automaton with state transitions then
  - A security policy is a statement that partitions the states of a system into a set of authorized or secure states and a set of unauthorized or non-secure states.
  - A secure system is a system that starts in an authorized state and cannot enter an unauthorized state.
  - A breach of security occurs when a system enters an unauthorized state.
- We expect a trusted system to enforce the required security policies.
Confidentiality, Integrity and Availability

- **Confidentiality**: Let $X$ be a set of entities and $I$ be some information. Then $I$ has the property of confidentiality with respect to $X$ is no member of $X$ can obtain information about $I$.

- **Integrity**: Let $X$ be a set of entities and $I$ some information or a resource. Then $I$ has the property of integrity with respect to $X$ if all members of $X$ trust $I$.

- **Availability**: Let $X$ be a set of entities and $I$ a resource. Then $I$ has the property of availability with respect to $X$ if all members of $X$ can access $I$. 
Elements of a Security Policy

- A security policy considers all relevant aspects of confidentiality, integrity and availability.
  - Confidentiality policy: Identifies information leakage and controls information flow.
  - Integrity Policy: Identifies authorized ways in which information may be altered. Enforces separation of duties.
  - Availability policy: Describes what services must be provided: example – a browser may download pages but no Java applets.
Reminder - Mechanism and Policy

- Example: University policy disallows cheating - copying another student’s homework assignment. Student A has her homework file world readable. Student B copies it. Who has violated policy?

- Mechanism should not be confused with policy.

- A security mechanism is an entity or procedure that enforces some part of a security policy.

- We have learnt some cryptographic and non-cryptographic mechanisms.
Types of Security Policies

- A military security policy (also called government security policy) is a security policy developed primarily to provide confidentiality.
  - Not worrying about trusting the object as much as disclosing the object
- A commercial security policy is a security policy developed primarily to provide integrity.
  - Focus on how much the object can be trusted.
- Also confidentiality policy and integrity policy.
CS Department Security Policy

- [http://cis.poly.edu/security-policy.html](http://cis.poly.edu/security-policy.html)
Security Models

- To formulate a security policy you have to describe entities it governs and what rules constitute it – a security model does just that!
- A security model is a model that represents a particular policy or set of policies. They are used to:
  - Describe or document a policy
  - Test a policy for completeness and consistency
  - Help conceptualize and design an implementation
  - Check whether an implementation meets requirements.
Relations and Orderings.

- For a set $S$, a relation $R$ is any subset of $S \times S$. For $(a, b)$ in $R$ we write $aRb$.
- A relation defined over $S$ is said to be:
  - Reflexive – if $aRa$ for all $a$ in $S$.
  - Transitive – If $aRb$ and $bRc$, then $aRc$. For $a, b, c$ in $S$.
  - Anti-symmetric – If $aRb$ and $bRa$, then $a = b$ for all $a, b$ in $S$.
- For $a, b$ in $S$, if there exists $u$ in $S$ such that $aRu$ and $bRu$, then $u$ is an upper bound of $a$ and $b$.
- Let $U$ be the set of upper bounds of $a$ and $b$. Let $u$ in $U$ such that there is no $t$ in $U$ where $tRu$. The $u$ is the least upper bound of $a$ and $b$.
- Similarly define lower bound and greatest lower bound.
Lattices

- A partial ordering occurs when a relation orders some, but not all, elements of a set.
- A lattice is a set of elements $S$ and a relation $R$ defined on the elements in $S$ such that
  - $R$ is reflexive, antisymmetric and transitive.
  - For every $s, t$ in $S$ there exists a lub.
  - For every $s, t$ in $S$ there exists a gub.
Examples

- The set \{0, 1, 2\} forms a lattice under the relation “less than equal to” i.e. \(\leq\)
- The set of integers form a lattice under the relation \(\leq\)
- Is B \(\leq\) G? Is B \(\leq\) E?
The Bell-La Padula (BLP) Model

- BLP model is a formal description of allowable paths of information flow in a secure system.
- Formalization of military security policy – confidentiality.
- Set of subjects $S$ and objects $O$. Each subject $s$ in $S$ and $o$ in $O$ has a fixed security class $L(s)$ (clearance) and $L(o)$ (classification).
- Security classes are ordered by a relation $\leq$
- Combines mandatory and discretionary access control.
A basic confidentiality classification system. The four levels are arranged on the list from most sensitive at top and least sensitive at bottom. In the middle are individuals grouped by their security clearance and at the right are documents grouped by their security level.

So Nasir cannot read personnel files and David can read any file. But what if Bud reads contents of personnel files and writes them on to a class file?
BLP – Simple Version

- Two properties characterize the secure flow of information:
  - Simple Security Property: A subject s may have read access to an object o if and only if $L(o) \leq L(s)$ and s has discretionary read access to o.
    (Security clearance of subject has to be at least as high as that of the object).
  - *-Property: A subject s who has read access to an object o may have write access to an object p only if $L(o) \leq L(p)$ and s has discretionary write access to o.
    (Contents of a sensitive object can only be written to objects at least as high. That is, prevent write-down).
Basic Security Theorem: Let $\Sigma$ be a system with a secure initial state $\sigma_0$ and let $T$ be a set of transformations. If every element of $T$ preserves the simple security property and $\ast$-property, then every state $\sigma_i$, $i \geq 0$, is secure.
BLP

- Divide each security level into a set of categories.
- Each security level and category forms a compartment. We say subjects have clearance for a set of compartments and objects being at the level of a compartment.
  - Need to know principle.
- Example: Let NUC, EUR and US be categories.
  - George is cleared for (TOP SECRET, {NUC, US})
  - A document may be classified as (CONFIDENTIAL, {EUR}).
Example Lattice

- The set of categories form a lattice under the subset \( \subseteq \) operation
We can now define a new relationship to capture the combination of security level and category set:

- \((L, C) \text{ dom } (L', C')\) if and only if \(L' \leq L\) and \(C' \subseteq C\).
- This relationship also induces a lattice on the set of compartments.

Example: George is cleared for \{SECRET, \{NUC, EUR\}\}, DocA is classified as \{CONFIDENTIAL, \{NUC\}\} DocB as \{SECRET< \{EUR, US\}\} and DocC as \{SECRET, \{EUR\}\}. George dom DocA and Doc C but not DocB.
Two properties characterize the secure flow of information:

Simple Security Property: A subject $s$ may have read access to an object $o$ if and only if $C(s) \text{ dom } C(o)$ and $s$ has discretionary read access to $o$.

*-Property: A subject $s$ who has read access to an object $o$ may have write access to an object $p$ only if $C(p) \text{ dom } C(o)$ and $s$ has discretionary write access to $o$.

Basic Security Theorem: Let $\Sigma$ be a system with a secure initial state $\sigma_0$ and let $T$ be a set of transformations. If every element of $T$ preserves the simple security property and *-property, then every state $\sigma_i$, $i \geq 0$, is secure.
Commercial Environments

- Commercial requirements differ from military requirements in their emphasis on preserving data integrity. For Example:
  1. Users will not write their own programs, but will use existing production programs and databases.
  2. Programmers will develop and test programs on a non-production system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
  3. A special process must be followed to install a program from the development system onto the production system.
  4. The special process in 3, above, must be controlled and audited.
  5. The management and auditors must have access to both the system state and to the system logs that are generated.
Principles of Operation

- *Separation of duty*. If two or more steps are required to perform a critical function, at least two different people should perform the steps.

- *Separation of function*. Developers do not develop new programs on production systems because of the potential threat to production data.

- *Auditing*. Auditing is the process of analyzing systems to determine what actions took place and who performed them. Commercial systems emphasize recovery and account-ability.
Biba Integrity Model

- Biba integrity model is counterpart (dual) of BLP model.
- It identifies paths that could lead to inappropriate modification of data as opposed to inappropriate disclosure in the BLP model.
- A system consists of a set $S$ of subjects, a set $O$ of objects, and a set $I$ of integrity levels. The levels are ordered.
- Subjects and Objects are ordered by the integrity classification scheme; denoted by $I(s)$ and $I(o)$.
Biba Integrity Model

- The properties of the Biba Integrity Model are:
  - Simple Integrity Property: Subject $s$ can modify (have write access to) object $o$ if and only if $I(s) \geq I(o)$.
  - Integrity *-property: If subject $S$ has read access to object $o$ with integrity level $I(o)$, $S$ can have write access to $p$ if and only if $I(o) \geq I(p)$. 
In commercial environment we worry about the integrity of the data in the system and the actions performed upon that data.

The data is said to be *in a consistent state* (or *consistent*) if it satisfies given properties.

For example, let $D$ be the amount of money deposited so far today, $W$ the amount of money withdrawn so far today, $YB$ be the amount of money in all accounts at the end of yesterday, and $TB$ be the amount of money in all accounts so far today. Then the consistency property is:

$$D + YB - W = TB$$
CW Model

- A well-formed transaction is a series of operations that leave the data in a consistent state if the data is in a consistent state when the transaction begins.
- The principle of separation of duty requires the certifier and the implementers be different people.
  - In order for the transaction to corrupt the data (either by illicitly changing data or by leaving the data in an inconsistent state), either two different people must make similar mistakes or collude to certify the well-formed transaction as correct.
The Clark-Wilson Model defines data subject to its integrity controls as *constrained data items* or CDIs.

Data not subject to the integrity controls are called *unconstrained data items*, or UDIs.

*Integrity verification procedures*, or IVPs, test that the CDIs conform to the integrity constraints at the time the IVPs are run. In this case, the system is said to be in a *valid state*.

*Transformation procedures*, or TPs, change the state of the data in the system from one valid state to another; TPs implement well-formed transactions.
CW Model

- **Certification Rule 1 (CR1):** When any IVP is run, it must ensure that all CDIs are in a valid state.

- **Certification Rule 2 (CR2):** For some associated set of CDIs, a TP must transform those CDIs in a valid state into a (possibly different) valid state.
  - CR2 defines a relation *certified C* that associates a set of CDIs with a particular TP;

- **Enforcement Rule 1 (ER1):** The system must maintain the *certified* relations, and must ensure that only TPs certified to run on a CDI manipulate that CDI.
CW Model

- **Enforcement Rule 2 (ER2):** The system must associate a user with each TP and set of CDIs. The TP may access those CDIs on behalf of the associated user. If the user is not associated with a particular TP and CDI, then the TP cannot access that CDI on behalf of that user.

- This defines a set of triple \((user, TP, \{ CDI \text{ set } \})\) to capture the association of users, TPs, and CDIs. Call this relation *allowed A*. Of course, these relations must be certified:
**CW Model**

- **Enforcement Rule 3 (ER3):** The system must authenticate each user attempting to execute a TP.

- **Certification Rule 4 (CR4):** All TPs must append enough information to reconstruct the operation to an append-only CDI.

- **Certification Rule 5 (CR5):** Any TP that takes as input a UDI may perform only valid transformations, for all possible values of the UDI. The transformation either rejects the UDI or transforms it into a CDI.

- **Enforcement Rule 4 (ER4):** Only the certifier of a TP may change the list of entities associated with that TP. No certifier of a TP, or of an entity associated with that TP, may ever have execute permission with respect to that entity.
The Chinese Wall Model is a model of a security policy that speaks equally to confidentiality and integrity. It describes policies that involve a conflict of interest in Business. For example:

In the environment of a stock exchange or investment house the goal of the model is to prevent a conflict of interest in which a trader represents two clients, and the best interests of the clients conflict, so the trader could help one gain at the expense of the other.
Chinese Wall Model

- The objects of the database are items of information related to a company.
- A company dataset (CD) contains objects related to a single company.
- A conflict of interest class (COI) contains the datasets of companies in competition.
- COI(O) represents the conflict of interest class that contains object O.
- CD(O) represents the company dataset that contains object O. The model assumes that each object belongs to exactly one conflict of interest class.
Chinese Wall Model

Anthony has access to the objects in the CD of Bank of America. Because the CD of Citibank is in the same COI as that of Bank of America, Anthony cannot gain access to the objects in Citibank’s CD. Thus, this structure of the database provides the required ability.
Chinese Wall Model

- Let PR(S) be the set of objects that S has read:
- **CW-simple security rule**: S can read O if and only if either:
  - There exists an object O’ such that CD(S) = CD(O’) and CD(O’’) = CD(O); or
  - For all objects O’, O’ ∈ PR(S) ⇒ COI(O’) ≠ COI(O).
- Hence the minimum number of subjects needed to access every object in a COI is the same as the number of CDs in that COI.
Chinese Wall Model

- In practice, companies have information they can release publicly, such as annual stockholders’ reports or filings before government commissions.
- Hence, \textit{CW-simple security rule}: \( S \) can read \( O \) if and only if either:
  - There exists an object \( O' \) such that \( CD(S) = CD(O') \) and \( CD(O') = CD(O) \); or
  - For all objects \( O', O' \)\( PR(S) \Rightarrow COI(O') \Rightarrow COI(O) \).
- \( O \) is a sanitized object
Chinese Wall Model

- Suppose Anthony and Susan work in the same trading house. Anthony can read objects in Bank of America’s CD, and Susan can read objects in Citibank’s CD. Both can read objects in ARCO’s CD. If Anthony can also write objects in ARCO’s CD, then he can read information from objects in Bank of America’s CD, write it to objects in ARCO’s CD, and then Susan can read that information;

- \textbf{CW-* Property Rule}: A subject $S$ may write to an object $O$ if and only if all of the following conditions hold:
  - The CW-simple security rule permits $S$ to read $O$; and
  - For all unsanitized objects $O'$, $S$ can read $O' \Rightarrow \text{CD}(O') = \text{CD}(O)$. 
A General Question

- Given a computer system, how can we determine if it is secure? More simply, is there a generic algorithm that allows us to determine whether a computer system is secure?

- What policy shall define “secure?” For a general result, the definition should be as broad as possible – access control matrix with some basic operations and commands.
Formalizing the question

- When a generic right \( r \) is added to an element of the access control matrix not already containing \( r \), that right is said to be leaked.

- Let a computer system begin in protection state \( s_0 \). If a system can never enter leak the right \( r \), the system (including the initial state \( s_0 \) ) is called safe with respect to the right \( r \). If the system can enter an unauthorized state, it is called unsafe with respect to the right \( r \).

- Our question (called the safety question): Does there exist an algorithm to determine whether a given protection system with initial state \( s_0 \) is safe with respect to a generic right \( r \)?
Fundamental Results of Security

- There exists an algorithm that will determine whether a given mono-operational protection system with initial state $s_0$ is safe with respect to a generic right $r$.
  - By enumerating all possible states we determine whether the system is safe. It is computationally infeasible, (NP-complete) but still it can be done in principle.
- Unfortunately, this result does not generalize to all protection systems.
Fundamental Results of Security

- It is undecidable whether a given state of a given protection system is safe for a given generic right.
  - For example, the protection system of the Unix OS, requires more than one command to be represented by the model used. Hence it is undecidable whether it is secure!