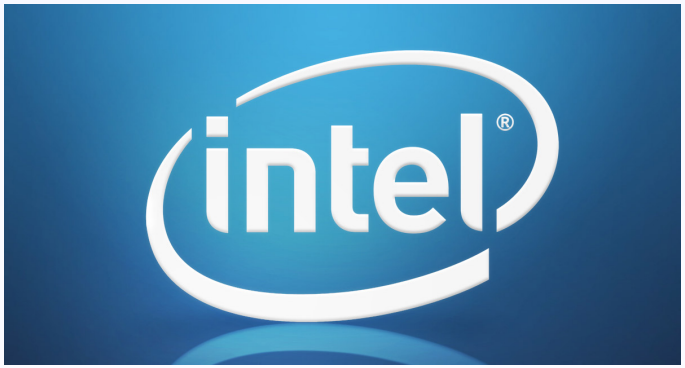
Constructing Trusted Code Base XII
Overview of security models

Aleksy Schubert & Jacek Chrząszcz



Security models

- Security models are fixed schemes of policies related to security
- There are different bases for security models
  - models of access rights
  - models of computation
  - models of distributed computing
  - etc.
- They define *trust* in technical terms
- They result in requirements on the code design and properties
- The requirements can be further verified either by experts or by tools
- A list of most popular models follows

Discretionary access control

- a way to restrict access to objects
- based on the identity of the subject and/or its group
- a subject may pass its permission (maybe indirectly) o to another subject (maybe under some conditions)
- example: Unix file system permissions

Mandatory access control

- a way to restrict access to objects
- operating system constrains the ability of a subject or initiator to access an object/perform operation
- subjects and objects are associated with their security attributes
- kernel examines the attributes and decides whether the access can take place
- examples: SELinux (in Linux), Mandatory Integrity Control (in Windows Vista)

Mult-level security (MLS)

- information with multiple incompatible classifications (i.e. at different security levels)
- permit access by users with different security clearances and needs-to-know, and
- prevent users from obtaining access to information for which they lack authorization

Multiple single-level

- separation of different levels of data through separate computers or virtual machines for each level
- cheap version of multilevel security
- not need for special changes to the OS or applications
- only extra machines required
- problems with combining information from different levels

Lattice-based access control

- involves objects (e.g. resources, computers, and applications) and subjects (e.g. individuals, groups or organizations).
- label-based mandatory access control model
- a lattice is used to define the levels of security
- for objects and subjects
- subject is only allowed to access an object if the security level of the subject is greater than or equal to that of the object (see Biba & Bell–LaPadula models)

Access control matrix

- relations among: subjects, objects, operations
- array of which subjects are granted which access to which objects
- entry: a list of operations allowed
- properties to verify:
  - each operation is guarded by a permission check
  - each operating subject has an identity
  - rights are not mutable
  - subject's identities are not mutable

Access control list (ACL)

- relations among: users, system processes, objects, operations
- which users or system processes are granted access to objects
- a list of permissions attached to an object
- what operations are allowed on given objects
- entry: a subject + an operation
- used in: filesystems, networking, sql
- properties to verify:
  - each operation is guarded by a permission check
  - each operating entity has an identity
  - ACLs are not mutable
  - user, process, object identities are not mutable
  - (the two above with small exceptions)

Object-capability model

- capability (also: key) is a communicable, unforgeable token of authority
- in other words: a reference with an associated set of access rights
- access to an object only through capabilities
- programs directly share capabilities with each other according to the principle of least privilege, and to the operating system infrastructure necessary to make such transactions efficient and secure. Capability-based security is to be contrasted with an approach that uses hierarchical protection domains.
- examples: L4 microkernel, Amoeba distributed operating system

Role-based access control (RBAC)

- permissions to perform certain operations are assigned to specific roles (not users)
- users are assigned particular roles
- management of individual user rights becomes a matter of simply assigning appropriate roles to the user's account
- rules
  - permissions only after assignment of a role
  - a subject's active role must be authorized for the subject
  - only permissions authorized for the role can be exploited
- possible other constraints and hierarchical arrangement of roles
- possible lattice-based access control (LBAC)

Role-based access control (RBAC)

- properties
  - A subject can have multiple roles.
  - A role can have multiple subjects.
  - A role can have many permissions.
  - A permission can be assigned to many roles.
  - An operation can be assigned many permissions.
  - A permission can be assigned to many operations.
- A subject may have multiple simultaneous sessions with different permissions.
- examples: Oracle DBMS, PostgreSQL 8.1, SELinux

Biba model

- relations among: users, data
- a level attached to each user and data item
- rules:
  - a subject at a level n must not read an object at level m s.t. $m < n$
  - a subject at a level n must not write an object at level m s.t. $m > n$
  - a process at level n cannot request access to subjects at a level m s.t. $m > n$
- properties to verify:
  - each read/write is guarded by a level check
  - each operating entity has a level
  - each piece of data has a level
  - levels are not mutable
  - (the above with small exceptions)

Bell–LaPadula model

- relations among: users, data
- a level attached to each user (clearance level) and data item (classification level)
- rules:
  - a subject at a level n must not read an object at level m s.t. $m > n$
  - a subject at a level n must not write to an object at level m s.t. $m < n$
  - discretionary access control through access matrix
- properties to verify (as in Biba model)

Non-interference (security)

- inputs and outputs of a system are classified as either low (open) or high (sensitive)
- non-interference property: any sequence of low inputs will produce the same low outputs, independent of the values in the high level inputs
- in patterns:

$$\forall M_1, M_2 : M_1 =_L M_2 \wedge (P, M_1) \rightarrow^* M_1' \wedge (P, M_2) \rightarrow^* M_2' \Rightarrow M_1' =_L M_2'$$

Brewer and Nash model (Chinese wall model)

- relations between: users, data, datasets, conflict of interest classes
- datasets are in conflict one with another
- people may only read data that is not in conflict with what they already posess

Brewer and Nash model (Chinese wall model)

- rules:
  - if any two objects o_1 and o_2 belong to the same dataset then they also belong to the same conflict of interest class
  - access (read or write) by a user u to an object o is possible only when conflict of interest class of o is different from all conflict of interest classes of objects already read or when dataset of o is equal to dataset of seen objects
  - initial state (no dataset is seen) is secure
  - if a user u has not seen anything than it can be granted access to any data
  - only one dataset in the conflict class of sanitized information
  - write to o_b by s_u is permitted if and only if when the access is logged and there is no object o_a (accessed by u) which can be read by s_u for which dataset of o_a is different than the dataset of o_b and the dataset of sanitized information
  - users may execute only processes they are allowed to
  - the processes may only access objects they are allowed to

Clark–Wilson model

- relations between: users and data
- users through well-formed transacions operate on data
- formal notions:
  - Constrained Data Item (CDI)
  - Unconstrained Data Item (UDI), aka input
  - Integrity Verification Procedure
  - Transformation Procedures (TPs), aka transactions
- operation:
  - IVP ensures that all CDIs in the system are valid
  - TPs enforce the integrity policy
  - A TP takes as input a CDI and/or UDI and produces a CDI
  - A TP transitions the system from one valid state to another valid state.
  - A TP must guarantee (via certification) that it transforms all possible values of a UDI to 'safe' CDIs

Clark–Wilson model

- two sets of rules: Certification Rules (C) and Enforcement Rules (E).
- Certification rules:
  1. IVP must ensure through its execution that the CDIs are valid.
  2. TP may only transform CDIs from valid to valid.
  3. Allowed relations (triples (user, TP, CDIs)) must obey the requirement of *separation of duty* (certifier and the implementer are different entities).
  4. TPs log enough information to reconstruct their operation.
  5. TP with UDIs as input perform only valid transactions for all possible UDIs (accept and convert to CDI or reject).

Clark–Wilson model

- Enforcement rules:
  1. A list of certified allowed relations is kept and execution is possible only in accordance wih the list.
  2. The certified allowed relations ensure that: only allowed users run associated certified TPs for appropriate valid CDIs to change them into valid CDIs.
  3. Users are authenticated on access to TPs (per TP request, not per session)
  4. The certifier of a TP is the only entity allowed to change the list of entities associated with the TP.

Graham-Denning model

- relations between: subjects, objects, access rights
- objects: system entities that need protection (files, processes etc.)
- subjects: system entities that can access objects (user, process etc.)
- access rights: kinds of access (read, write, execure, being owner)
- realised as a matrix between subjects and objects with sets of rights in entries

Graham-Denning model

- there are 8 protection rules that describe
  1. How to securely create an object
  2. How to securely create a subject
  3. How to securely delete an object
  4. How to securely delete a subject
  5. How to securely provide the read access right
  6. How to securely provide the grant access right
  7. How to securely provide the delete access right
  8. How to securely provide the transfer access right
- each object has an *owner* (subject) that has special rights on it
- each subject has a *controller* (subject) that has special rights on it
- each rule is associated with a precondition (e.g. if x wants to delete o, it must be its owner)

Harrison-Ruzzo-Ullman (HRU)

- refinement of the Graham-Denning model
- relations between sets of: subjects, objects, generic rights and commands C
- configurations: (S, O, P)
  - S: current subjects,
  - O: current objects, and
  - P: access matrix
- subjects are required to be part of the objects
- entry for $[s, o]$ is a subset of the generic rights
- commands are composed of primitive operations
- commands have a list of pre-conditions that require certain rights to be present for a pair $[s, o]$
- commands work in transactional manner (failure inside causes rollback)

Harrison-Ruzzo-Ullman (HRU)

- primitive operations can modify the access matrix:
  - adding or removing access rights
  - adding or removing subjects
  - adding or removing objects.
- study of computational properties of security models (reachability is undecidable)

High/low-water mark

- relation between: objects, users and security levels
- rules:
  - users can open any object at a security level less than theirs
  - the object is relabeled to reflect the highest security level currently open
- gradual movement of all objects towards the highest security level in the system
- example: if a user is assigned to assemble the daily intelligence briefing at the TOP SECRET level, refers to a dictionary for spellchecking it makes the dictionary TOP SECRET
- low-water mark: corresponds to the Biba model
  - write down is permitted
  - the subject level is degraded to the object level
- introduced in Clark Weissmann in 1969 (pre-dates Bell-LaPadula security model from 1972)

Take-grant protection model

- relation between: subjects and objects
- relation modelled as a graph:
  - verices: subjects and objects
  - directed edges: rights of the source over the destination
- special rights: take, grant
- take and grant constrain graph rewriting rules with the principles
  - *take rule* allows a subject to take rights of another object (add an edge originating at the subject)
  - *grant rule* allows a subject to grant own rights to another object (add an edge terminating at the subject)
  - *create rule* allows a subject to create new objects (add a vertex and an edge from the subject to the new vertex)
  - *remove rule* allows a subject to remove rights it has over on another object (remove an edge originating at the subject)

Take-grant protection model

- Preconditions
  - for take(o,p,r):
    - subject s has the right Take for o
    - object o has the right r on p
  - for grant(o,p,r):
    - subject s has the right Grant for o
    - s has the right r on p

Type enforcement

- relation between: subjects, objects and security contexts of a domain
- a security context in a domain is defined by a domain security policy
- control over: process execution, domain transition and authorization scheme
- evaluation of rules from the source security context of a subject, against a set of rules from the target security context of the object
- clearance decision depends on TE access description
- labels are attached to subject and object:
  - a *domain label* for a subject
  - a *type label* for an object