CIS 6930:
IoT Security

Lecture 7

Prof. Kaushal Kafle

Spring 2025

Derived from slides by Adwait Nadkarni, William Enck, Micah Sherr and Patrick McDaniel

Class Notes and Clarifications

- **Related work due today!**
- Midterm grades
  - Based on:
    - Reviews
    - Class Participation
    - Project milestones (so far)
  - Presentations will count only for the final grades
- Paper reviews
  - First two reviews: 10% weight
  - Reviews after that count for full
- Project related questions:
  - Utilize TA office hours

Access Control Administration

There are two central ways to specify a policy

- *Discretionary* - object "owners" define policy
  - Users have discretion over who has access to what objects and when (trusted users)
  - Canonical example: the UNIX filesystem
    - RWX assigned by file owners
- *Mandatory* - Environment enforces static policy
  - Access control policy defined by environment, user has no control over access control (untrusted users)
  - Canonical example: process labeling
    - System assigns labels for processes, objects, and a dominance calculus is used to evaluate rights

DAC vs. MAC

- **<u>Discretionary</u> Access Control**
  - User defines the access policy
  - Can pass rights onto other subjects (called *delegation*)
  - Their programs can pass their rights
    - Consider a Trojan horse
- **<u>Mandatory</u> Access Control**
  - System defines access policy
  - Subjects cannot pass rights
  - Subjects' programs cannot pass rights
    - Consider a Trojan horse here

DAC vs. MAC in Access Matrix

- Subjects:
  - DAC: users
  - MAC: labels
- Objects:
  - DAC: files, sockets, etc.
  - MAC: labels
- Operations:
  - Same
- Administration:
  - DAC: owner, copy flag, …
  - MAC: external, reboot
- MAC: largely static matrix;
- DAC: all can change

	O_1	O_2	O_3
S_1	Y	Y	N
S_2	N	Y	N
S_3	N	Y	Y

Safety Problem

- For a protection system
  - (ref mon, protection state, and administrative operations)
- Prove that any future state will not result in the leakage of an access right to an unauthorized user
  - Q: Why is this important?
- For most discretionary access control models,
  - Safety is *undecideable*
- Means that we need another way to prove safety
  - Restrict the model (no one uses)
  - Test incrementally (constraints)
- *How does the safety problem affect MAC models?*

Sandboxing

- An execution environment for programs that contains a limited set of rights
  - A subset of your permissions (meet secrecy and integrity goals)
  - Cannot be changed by the running program (mandatory)

Case Study – Android UIDs

- Android is a *Linux-based system*
- Apps are security principles*, treated as users*
- Apps acquire *permissions* to access ...
- What separates apps from one another?
- What separates Apps from the kernel?
- What prevents apps from access arbitrary storage?

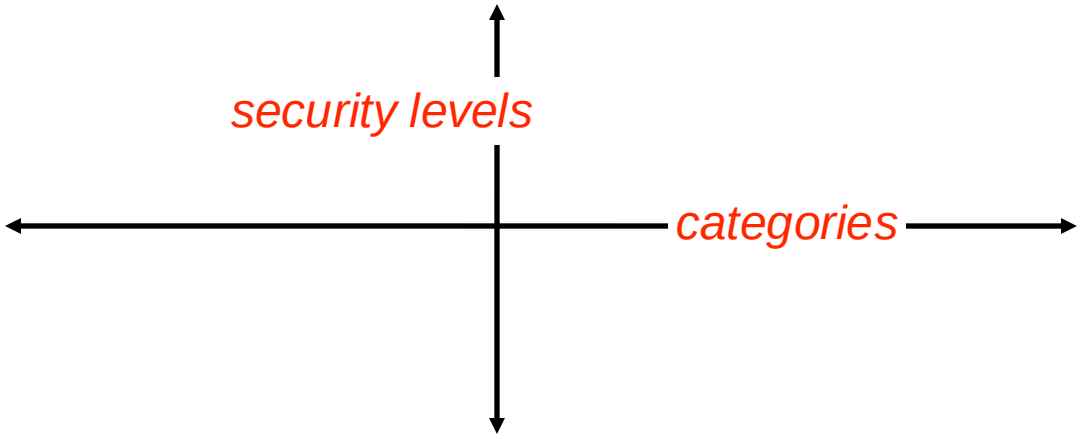
Access Control Models

- What language should I use to express policy?
  - Access Control Model
- Oodles of these
  - Some specialize in secrecy
    - Bell-LaPadula
  - Some specialize in integrity
    - Clark-Wilson
  - Some focus on jobs
    - RBAC
  - Some specialize in least privilege
    - SELinux Type Enforcement
- Q: Why are there so many different models?

Information Flow Control

Multilevel Security

- A multi-level security system tags all object and subject with security tags classifying them in terms of sensitivity/access level.
  - We formulate an access control policy based on these levels
  - We can also add other dimensions, called categories which horizontally partition the rights space (in a way similar to that as was done by roles)



US DoD Policy

- Used by the US military (and many others), the Lattice model uses MLS to define policy
- Levels:

UNCLASSIFIED < CONFIDENTIAL < SECRET < TOP SECRET

- Categories (actually unbounded set)

NUC(lear), INTEL(igence), CRYPTO(graphy)

- Note that these levels are used for physical documents in the governments as well.

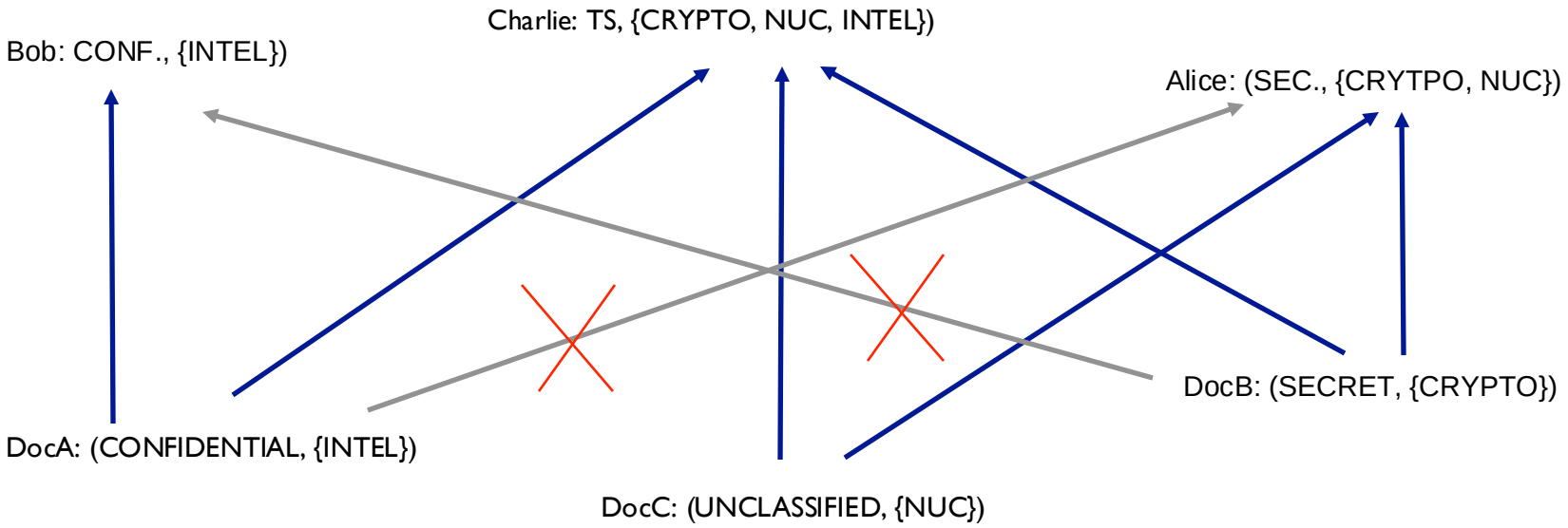
Assigning Security Levels

- All subjects are assigned clearance levels and compartments
  - Alice: (SECRET, {CRYTPO, NUC})
  - Bob: (CONFIDENTIAL, {INTEL})
  - Charlie: (TOP SECRET, {CRYPTO, NUC, INTEL})

- All objects are assigned an access class
  - DocA: (CONFIDENTIAL, {INTEL})
  - DocB: (SECRET, {CRYPTO})
  - DocC: (UNCLASSIFIED, {NUC})

Evaluating Policy

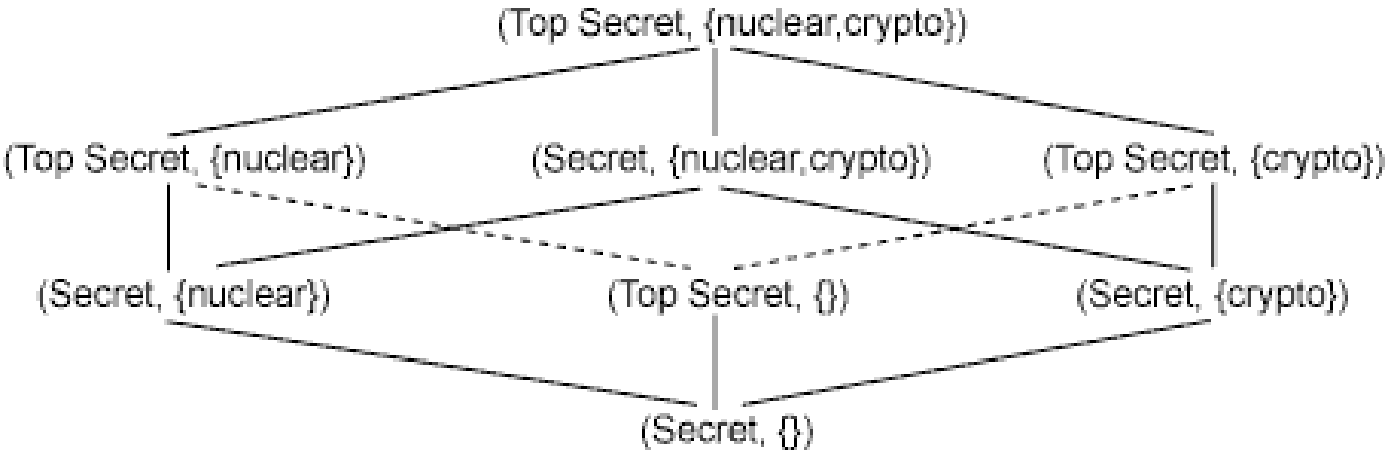
- Access is allowed if
- subject clearance level >= object sensitivity level *and* subject categories ⊇ object categories (*read down*)



Q: What would *write-up* be?

Bell-LaPadula (BLP) Model

- A Confidentiality MLS policy that enforces:
  - *Simple Security Policy*: a subject at specific classification level cannot read data with a higher classification level. This is short hand for "*no read up*".
  - ** (star) Property*: also known as the confinement property, states that subject at a specific classification cannot write data to a lower classification level. This is shorthand for "*no write down*".



How about integrity?

- MLS as presented before talks about who can "read" a document (confidentiality)
- Integrity considers who can "write" to a document
  - Thus, who can effect the integrity (content) of a document
  - Example: You may not care who can read DNS records, but you better care who writes to them!
- Biba defined a dual of secrecy for integrity
  - Lattice policy with, "no read down, no write up"
    - Users can only *create* content at or *below* their own integrity level (a monk may write a prayer book that can be read by commoners, but not one to be read by a high priest).
    - Users can only *view* content at or *above* their own integrity level (a monk may read a book written by the high priest, but may not read a pamphlet written by a lowly commoner).

Integrity, Sewage, and Wine

- Mix a gallon of sewage and one drop of wine gives you?
- Mix a gallon of wine and one drop of sewage gives you?

Integrity is really a contaminant problem: you want to make sure your data is not contaminated with data of lower integrity.

Biba (example)

- Which users can modify what documents?
  - Remember "*no read down, no write up*"

Charlie: (TS, {CRYPTO, NUC, INTEL})

Bob: (CONF., {INTEL})

Alice: (SEC., {CRYTPO, NUC})

?????

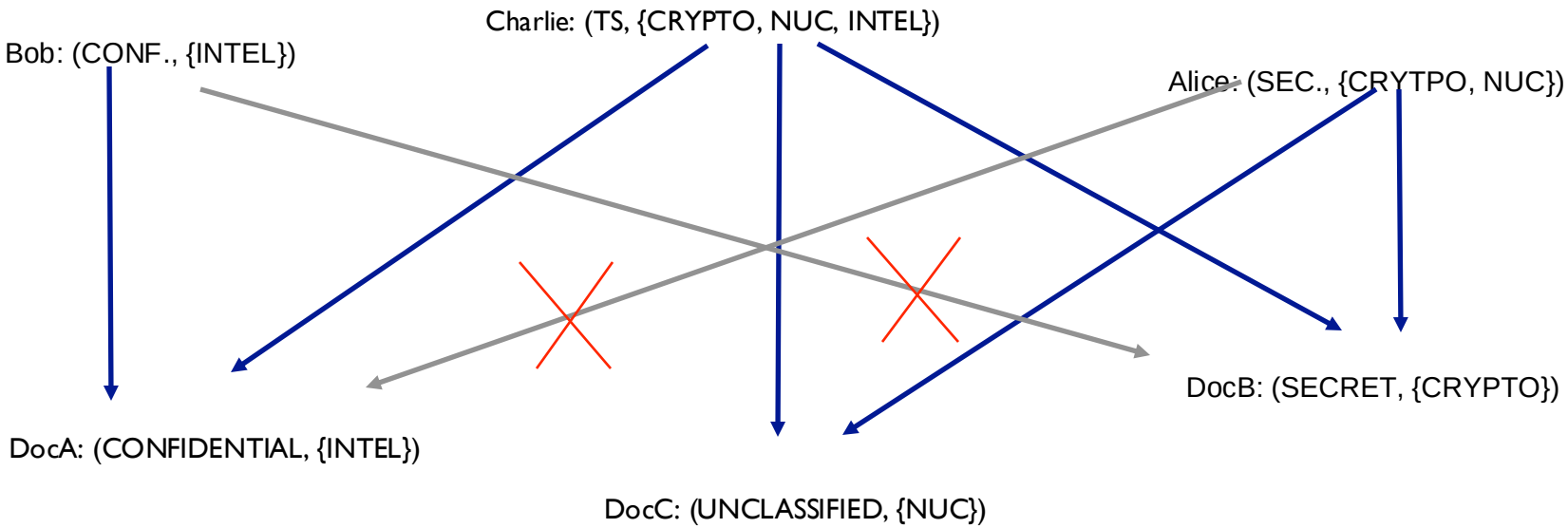
DocB: (SECRET, {CRYPTO})

DocA: (CONFIDENTIAL, {INTEL})

DocC: (UNCLASSIFIED, {NUC})

Biba (example)

- Which users can modify what documents?
  - Remember "*no read down, no write up*"

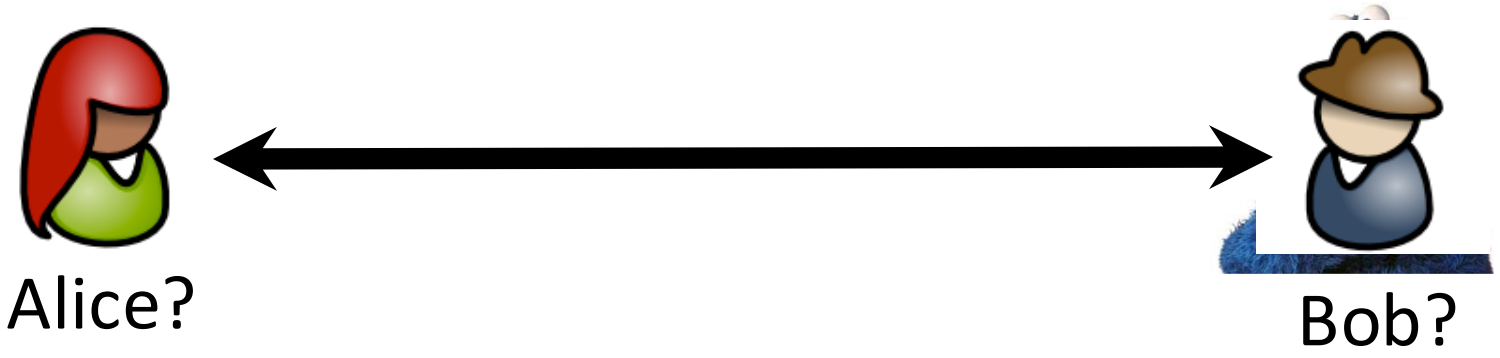


LOMAC

- Low-Water Mark integrity
  - Change integrity level based on actual dependencies
- Subject is initially at the highest integrity
  - But integrity level can change based on objects accessed
- Ultimately, subject has integrity of lowest object read
  - Example of "*self revocation*"

User Authentication


Username : admin
Password : admin



Authentication

What is Authentication?

- Short answer: establishes identity
  - Answers the question: To whom am I speaking?

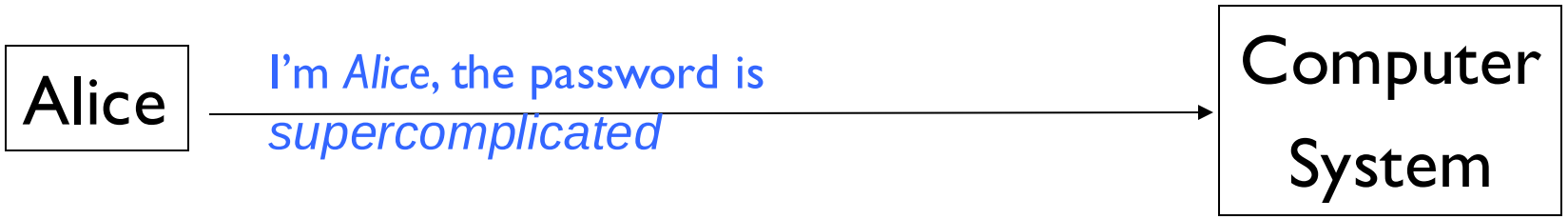
- Long answer: evaluates the authenticity of identity proving credentials.
  2 parts:
  - Credential – is proof of identity
  - Evaluation – process that assesses the correctness of the association between credential and claimed identity
    - for some purpose
    - under some policy (what constitutes a good cred.?)

Examples of Authentication

- Two broad types of authentication
  - User authentication
    - Allow a user to prove his/her identity to another entity (e.g., a system, a device).
  - Message authentication
    - Verify that a message has not been altered without proper authorization.

Authentication Mechanisms

- Password-based authentication
  - Use a secret quantity (the password) that the prover states to prove he/she knows it.
  - Threat: password guessing/dictionary attack



Authentication Mechanisms (Cont'd)

- Address-based authentication
  - Assume the identity of the source can be inferred based on the network address from which packets arrive (aka *authentication by assertion*)
  - Adopted early in UNIX and VMS
- Berkeley *rtools* (*rsh*, *rlogin*, etc)
  - */etc/hosts.equiv* file
    - List of computers
  - Per user *.rhosts* file
    - List of <computer, account>
- Threat
  - Spoof of network address
    - Not authentication of source addresses

Authentication Mechanisms (Cont'd)

- Cryptographic authentication protocols
  - Basic idea:
    - A prover proves some information by performing a cryptographic operation on a quantity that the verifier supplies.
  - Usually reduced to the knowledge of a secret value
    - A symmetric key
    - The private key of a public/private key pair

Why authentication?

- We live in a world of rights, permissions, and duties
  - Authentication establishes our identity so that we can obtain the set of rights
  - E.g., we establish our identity with a store by providing a valid credit card which gives us rights to purchase goods
    - this is a *physical* authentication system
    - ***Threats?***

Why authentication?

- Same in online world, just with different constraints
  - Vendor/customer are not physically co-located, so we must find other ways of providing identity
    - e.g., by providing credit card number ~ electronic authentication system
  - Risks (for customer and vendor) are different
    - Q: How so?
  - **Computer security is critically dependent on the proper design, management, and application of authentication systems**

What is Identity?

- That which gives you access (your credential) … which is largely determined by context
  - We all have lots of identities
  - Pseudo-identities
- Really, determined by who is evaluating credential
  - Driver's License, Passport, SSN prove …
  - Credit cards prove …
  - Signature proves …
  - Password proves …
  - Voice proves …

- ***Exercise***: Give an example of bad mapping between identity and the purpose for which it was used.