Computer Security (COM-301)

Mandatory Access Control
Confidentiality security models

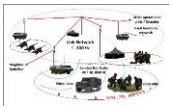
Carmela Troncoso
SPRING Lab
carmela.troncoso@epfl.ch

Some slides/ideas adapted from: George Danezis

We talked about DAC

MANDATORY ACCESS CONTROL (MAC)

Central security policy *assigns permissions*



DISCRETIONARY ACCESS CONTROL (DAC)

Object owners *assign permissions*

STRAVA

Theoretical lecture ahead!

Understand **basic concepts and principles of security** design and engineering that will **outlast current technology**

In a **DAC** owners have all powers on the objects they create: they have the right to establish permission access.
In a **MAC** the policy is given, and object creators cannot set their own permissions, they can only assign the rights dictated by this policy.

Mandatory Access Control

Access to and operations on resources **are determined** by the security policy

- "owner" may not exist or not have power to set permissions against policy
- the security policy **must** be enforced despite subjects trying to subvert it

In this lecture we will study what are the principles behind the main Mandatory Access Control security policy models.

In these policies, the policies **determine all** the relations between subjects and objects. It may be the case that there are no "object owners", i.e., even if a user creates an object she has no rights to establish the access policy of this object.

Security models

SECURITY MODEL: a **design pattern** for a specific security property or set of properties

When faced with a standard security problem → use well-known model!

A **security model** is not a policy or a mechanism itself. It is a design pattern, a way of reasoning about **security properties** in order to build security policies that ensure those properties.

Security models **only** provide the design patterns, but not the details of the policy! Those still need to be defined by the security engineer and if they are not defined carefully following the pattern does not guarantee security.

Security models

SECURITY MODEL: a **design pattern** for a specific security property or set of properties

When faced with a standard security problem → use well-known model!

The devil is in the details!

Many aspects not covered by the model!

- who are the subjects?
- what are the objects?
- what mechanisms to use to implement it?

A **security model** is not a policy or a mechanism itself. It is a design pattern, a way of reasoning about **security properties** in order to build security policies that ensure those properties.

Security models **only** provide the design patterns, but not the details of the policy! Those still need to be defined by the security engineer and if they are not defined carefully following the pattern does not guarantee security.

Bell-La Padula (BLP) model: Protecting confidentiality

Subjects **S** and objects **O** associated to a **level** of confidentiality

Subjects access rights are defined by four attributes:

Execute: the subject cannot see or modify the object, but can run it
Read: the subject can only see the object but cannot modify it
Append: the subject cannot read the object, but can add attach new content
Write: the subject can see the object and add content or modify existing content

These access rights are defined in an access control matrix

D. Bell and L. LaPadula. "Secure computer systems: Unified exposition and Multics interpretation". Technical Report ESD-TR-75-306, MITRE Corp., March 1976.

The Bell LaPadula model for confidentiality assumes the system has subjects **S** and objects **O.**

Reminder: *Confidentiality*: protect objects **O** access from non-authorized subjects **S**

Access to objects are given by four possible attributes. These attributes define the permissions the subject has on the object in two axis: can be observed (able to see the object or its content) and can be altered (able to change the object):

Execute: the subject cannot observe or modify the object, but has the capability to run
Read: the subject can only observe the object but cannot perform any modification (no addition, change, or deletion)
Append: the subject cannot read the object, but can add content to at the end without modifying any of the existing content
Write: the subject can see the object and can modify it (add, delete, change, any part of the object)

The system state at any point is defined as:

The level (i.e., the classification) of both subjects and objects with respect to confidentiality (defined in the next slides)

An access control matrix that assigns attributes for each subject with respect to each object

The current accesses (which subject is accessing with object with what permission) that are authorized at that point in time

Level function for objects: **Classification**

Objects are associated to a **Security Level**
(they have a **label**, and belong to one or more **categories**)

Security Level = (Classification, {set of categories})

Classification - total order of **labels** *(e.g., Unclassified, Confidential, Secret, Top Secret)*
Categories – compartments of objects with a common topic (*e.g., Nuclear, NATO, Crypto*)

The **Classification** of objects defines "how secret" objects are. Objects not only are associated a classification through the use of a **label**, but also a **Category** which confines it into a compartment.

The tuple (Classification, {set of categories}) is called a **Security Level**

Classification: dominance relationship

DOMINANCE RELATIONSHIP

A security level (l1,c1) "dominates" (l2, c2) *if and only if* l1 >= l2 **and** c2 is a subset of c1

Labels: **A**dmin < **N**urse < **S**urgeon < **D**octor
Categories: DEMOGRAPHICS, ANALYSIS, RESULTS

8

The idea behind dominance is that a level (l1,c1) dominates a level (l2,c2) if:

- the label is higher (i.e., the label l1 denotes more secrecy than the label l2)
- and the categories in c2 are a subset of c1 ← the categories in lower levels **always** allow access to less categories

(D, {}) dominates (S, {}) ← TRUE: Doctor is a higher classification than Surgeon

(S, {}) dominates (N, {RESULTS}) ← FALSE: Surgeon is a higher classification than Nurse, but this Nurse can see more than the Surgeon (The nurse can see RESULTS while the Surgeon cannot). The second condition is not fulfilled. There is no dominance relation between these categories

(S, {DEMOGRAPHICS,RESULTS}) dominates (N, {DEMOGRAPHICS}) ← YES: Surgeon is a higher classification than Nurse, and the Surgeon can see more information than the nurse

(D, {ANALYSIS,RESULTS}) dominates (S, {DEMOGRAPHICS}) ← FALSE: Doctor is a higher classification than Surgeon, but this Surgeon sees different information. The second condition is not fulfilled. There is no dominance relation between these categories

What level dominates them all? → The level (Doctor, {DEMOGRAPHICS, ANALYSIS, RESULTS}
What level dominates only itself? → The level (Admin,{})

Classification: dominance relationship

DOMINANCE RELATIONSHIP

A security level (l1,c1) "dominates" (l2, c2) *if and only if* l1 >= l2 **and** c2 is a subset of c1

Labels: **A**dmin < **N**urse < **S**urgeon < **D**octor
Categories: DEMOGRAPHICS, ANALYSIS, RESULTS

Which statements are true?

(D, {}) dominates (S, {})

(S, {}) dominates (N, {RESULTS})

(S, {DEMOGRAPHICS,RESULTS}) dominates (N, {DEMOGRAPHICS})

(D, {ANALYSIS,RESULTS}) dominates (S, {DEMOGRAPHICS})

What level dominates them all?
What level dominates only itself?

9

The idea behind dominance is that a level (l1,c1) dominates a level (l2,c2) if:

- the label is higher (i.e., the label l1 denotes more secrecy than the label l2)
- and the categories in c2 are a subset of c1 ← the categories in lower levels **always** allow access to less categories

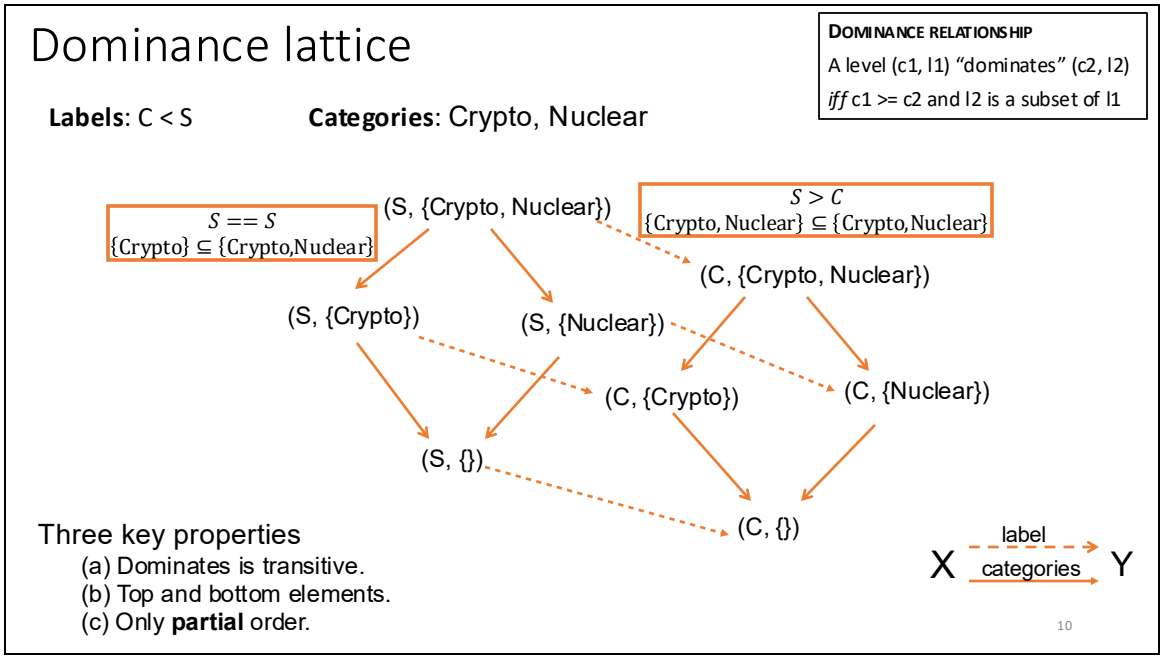
(D, {}) dominates (S, {}) ← TRUE: Doctor is a higher classification than Surgeon

(S, {}) dominates (N, {RESULTS}) ← FALSE: Surgeon is a higher classification than Nurse, but this Nurse can see more than the Surgeon (The nurse can see RESULTS while the Surgeon cannot). The second condition is not fulfilled. There is no dominance relation between these categories

(S, {DEMOGRAPHICS,RESULTS}) dominates (N, {DEMOGRAPHICS}) ← YES: Surgeon is a higher classification than Nurse, and the Surgeon can see more information than the nurse

(D, {ANALYSIS,RESULTS}) dominates (S, {DEMOGRAPHICS}) ← FALSE: Doctor is a higher classification than Surgeon, but this Surgeon sees different information. The second condition is not fulfilled. There is no dominance relation between these categories

What level dominates them all? → The level (Doctor, {DEMOGRAPHICS, ANALYSIS, RESULTS}
What level dominates only itself? → The level (Admin,{})



A way of reasoning about dominance is to draw the dominance lattice. It represents dominance relationships between Security levels on both aspects: labels and categories.

Note that dominance is transitive. If

(S,{CRYPTO, NUCLEAR}) dominates (S,{NUCLEAR}),
and (S,{NUCLEAR}) dominates (C,{NUCLEAR}),
then (S,{CRYPTO, NUCLEAR}) dominates (C,{NUCLEAR}).

As we saw in the previous slides, there is one level that dominates all the other (top) and one level that is dominated by all (bottom).

Also note that there is no total order between security levels. Some of them have no relationship!

(S,{NUCLEAR}) has no relation to (S, {CRYPTO})

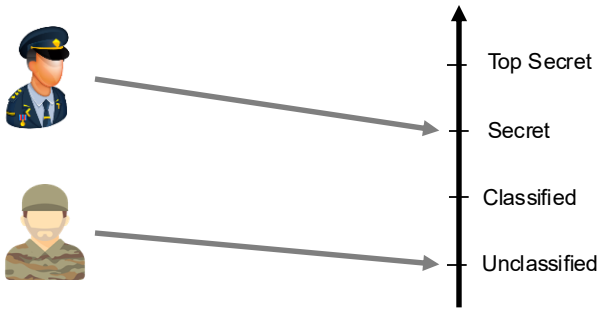
Level function for subjects: **Clearance level**

BLP calls this also “classification”

Clearance – maximum security level a subject has been assigned: ***clearance level(S_i)***

Current security level – subjects can operate at lower security levels: ***current-level(S_i)***

*level(S_i) **must** dominate current-level(S_i) !!!*



The **Clearance** for subjects defines “how secret” objects they can access. Besides their overall clearance, subjects can act with a lower level (called **current level).**

BLP System: ss-property

SIMPLE SECURITY PROPERTY (SS-PROPERTY)

If (subject, object, r) is a current access, then level(subject) dominates level(object)

CLEARANCE: SECRET

OBJECTS CLASSIFICATION

- Top Secret
- Secret
- Classified
- Unclassified

12

The **ss-property** fundamentally says that a subject can only access objects whose classification are dominated by the clearance of the subject. In layman terms, the subject can read objects with lower classification than its clearance.

This property is also known as No Read Up, as it says that subjects cannot read anything above their Clearance.

This general, whose Clearance is secret, can Access documents that are Secret, Classified, and Unclassified, but not Top Secret.

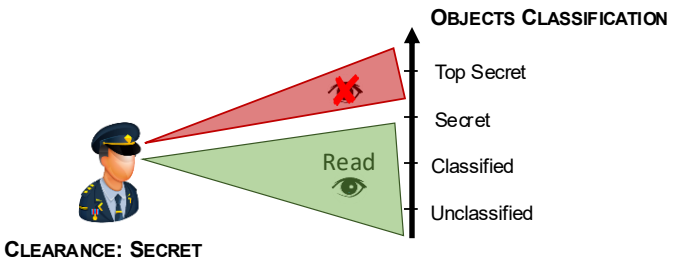
[Note that for simplicity in this slide (and the next ones) we skip the categories, but the dominance relationship would apply]

BLP System: ss-property

SIMPLE SECURITY PROPERTY (SS-PROPERTY)

If (subject, object, r) is a current access, then level(subject) dominates level(object)

No Read Up (NRU)



13

The **ss-property** fundamentally says that a subject can only access objects whose classification are dominated by the clearance of the subject. In layman terms, the subject can read objects with lower classification than its clearance.

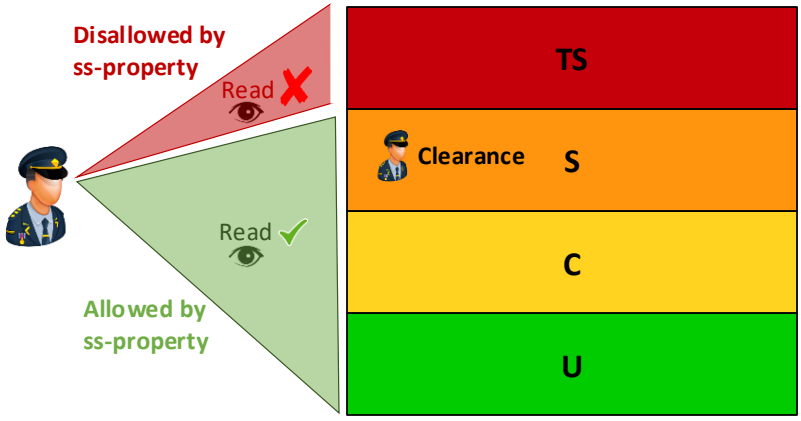
This property is also known as No Read Up, as it says that subjects cannot read anything above their Clearance.

This general, whose Clearance is secret, can Access documents that are Secret, Classified, and Unclassified, but not Top Secret.

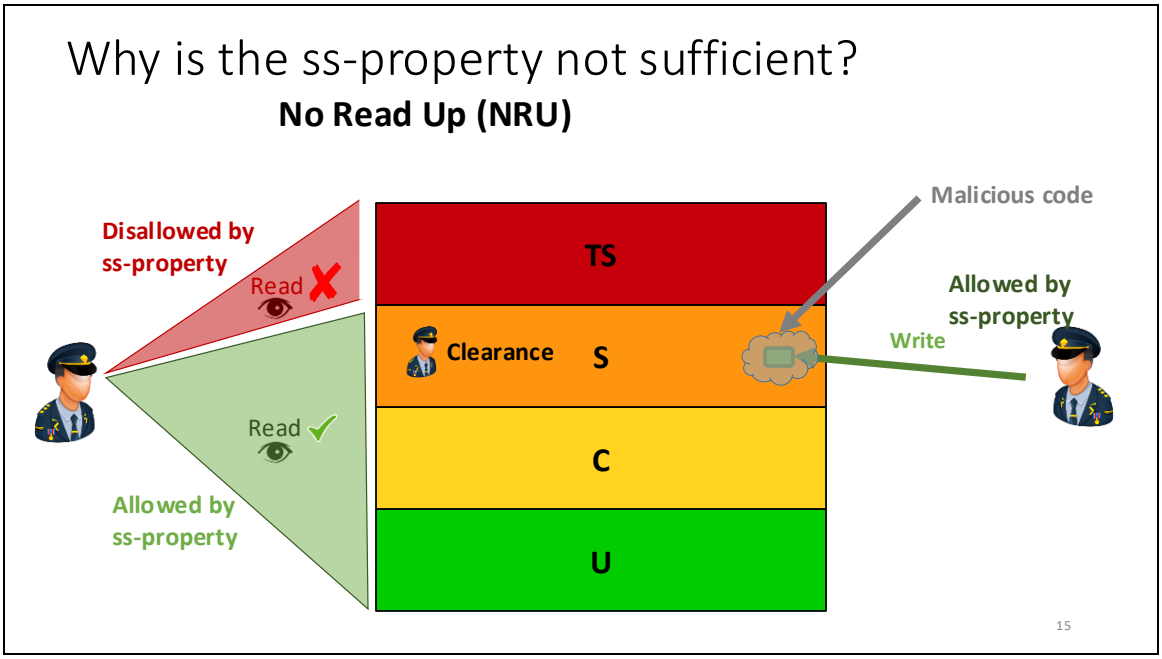
[Note that for simplicity in this slide (and the next ones) we skip the categories, but the dominance relationship would apply]

Why is the ss-property not sufficient?

No Read Up (NRU)



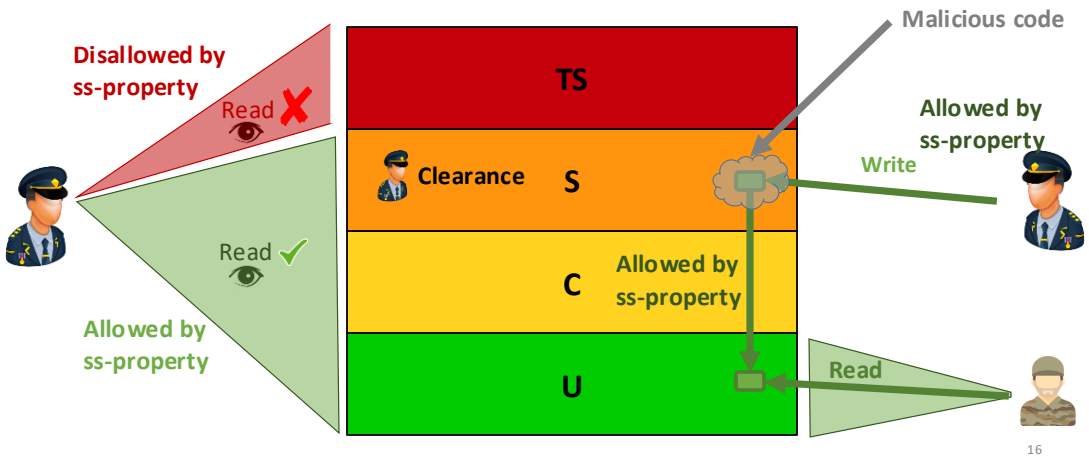
The problem with just forbidding people to not read up is that, if you have a malicious user that can read high level classified information, it can copy this information to a lower level. Then, subjects with less Clearance, who by the ss-property could not read this object now are actually allowed to read.



The problem with just forbidding people to not read up is that, if you have a malicious user that can read high level classified information, it can copy this information to a lower level. Then, subjects with less Clearance, who by the ss-property could not read this object now are actually allowed to read.

Why is the ss-property not sufficient?

No Read Up (NRU)



The problem with just forbidding people to not read up is that, if you have a malicious user that can read high level classified information, it can copy this information to a lower level. Then, subjects with less Clearance, who by the ss-property could not read this object now are actually allowed to read.

BLP System: *-property

STAR PROPERTY (*-PROPERTY)

if a subject has simultaneous "observe" (r,w) access to O_1
and "alter" (a,w) access to O_2 then level (O_2) dominates level (O_1)

CLEARANCE: SECRET

OBJECTS CLASSIFICATION

- Top Secret
- Secret
- Classified
- Unclassified

17

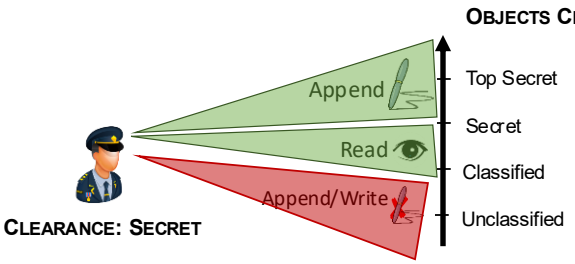
To solve this issue, BLP has a second property: ***-property**. This property says that if a subject can observe an object at a Classification level, then this subject can only modify object in higher levels. Note that, since the Clearance of this subject will not allow them to read those levels (ss-property, No-Read-Up), this subject can only *append* to higher levels.

In other words, what this property says is that subjects cannot write in clearance levels lower than theirs, so that they cannot leak information.

BLP System: *-property

STAR PROPERTY (*-PROPERTY)

if a subject has simultaneous "observe" (r,w) access to O_1
and "alter" (a,w) access to O_2 then level (O_2) dominates level (O_1)



No Write Down (NWD)

18

To solve this issue, BLP has a second property: ***-property**. This property says that if a subject can observe an object at a Classification level, then this subject can only modify object in higher levels. Note that, since the Clearance of this subject will not allow them to read those levels (ss-property, No-Read-Up), this subject can only *append* to higher levels.

In other words, what this property says is that subjects cannot write in clearance levels lower than theirs, so that they cannot leak information.

BLP System: ds-property

DISCRETIONARY PROPERTY (DS-PROPERTY)

if an access (subject, object, action) takes place it must be in the access control matrix

Information should only be accessed on a "need-to-know" basis

MAC needs **DAC** (least privilege inside the Security Level)

Also important to protect integrity

Regardless of the subjects clearance levels, information should only be known on a need to know basis! (Least privilege principle)

Thus, the system has an access control matrix that establishes which subjects can access with objects (provided that the access is permitted by the MAC policy, of course).

Using the DAC, the system can also protect the integrity of objects by forbidding subjects from writing on them.

BLP: Basic Security Theorem

BASIC SECURITY THEOREM

if all state transitions are secure, and the initial state is secure, then every subsequent state is secure regardless of the inputs

If for any individual access:

(1) the ss-property holds.

(2) the *-property holds.

(3) the ds-property holds.

… then for any sequential composition security holds!

A system can be analyzed in terms of single step transitions of states!!

20

With these three properties, one can reason about the security of the system.

Every time there is a state transition (recall that a state is the current access permissions, the access control matrix, and the cleareance/classification): one can decide whether the new state is secure.

But… these properties are not enough

Assume

- level(s_1) is TS (Top Secret)
- level(s_2) is C (Confidential)

Sequence of events

1) s_2 creates o_2 → level(o_2) = C

Even if the rules are respected, there can still be information leakage from higher levels to lower levels:

- s2 creates o2 – This is allowed, s2 is not writing down. The level of o2 is C.
- s1 sets his current level as C and reads o2 in C – This is allowed, s1 is not reading up.
- s1 may change the level of o2 – This is allowed, BLP says **nothing** about changing objects' level
- s2 attempts to access o2 in C – This is allowed, s2 can read in C because it is s2's clearance level

The fact that the file is still there when s2 tries to access it can leak one bit of information (e.g., s1 tells s2 that when the troops will start moving he will make o2 disappear)

But… these properties are not enough

Assume

- level(s_1) is TS (Top Secret)
- level(s_2) is C (Confidential)

Sequence of events

1) s_2 creates o_2 → level(o_2) = C
2) s_1 reads C and either:
    - changes the object level → level(o_2) = TS
    - leaves object level untouched → level(o_2) = C

22

Even if the rules are respected, there can still be information leakage from higher levels to lower levels:

- s2 creates o2 – This is allowed, s2 is not writing down. The level of o2 is C.
- s1 sets his current level as C and reads o2 in C – This is allowed, s1 is not reading up.
- s1 may change the level of o2 – This is allowed, BLP says **nothing** about changing objects' level
- s2 attempts to access o2 in C – This is allowed, s2 can read in C because it is s2's clearance level

The fact that the file is still there when s2 tries to access it can leak one bit of information (e.g., s1 tells s2 that when the troops will start moving he will make o2 disappear)

But… these properties are not enough

Assume

- level(s_1) is TS (Top Secret)
- level(s_2) is C (Confidential)

Sequence of events

1) s_2 creates o_2 → level(o_2) = C
2) s_1 reads C and either:
    - changes the object level → level(o_2) = TS
    - leaves object level untouched → level(o_2) = C
3) s_2 attempts to access to o_2 in C → success or failure leaks 1 bit of information!

23

Even if the rules are respected, there can still be information leakage from higher levels to lower levels:

- s2 creates o2 – This is allowed, s2 is not writing down. The level of o2 is C.
- s1 sets his current level as C and reads o2 in C – This is allowed, s1 is not reading up.
- s1 may change the level of o2 – This is allowed, BLP says **nothing** about changing objects' level
- s2 attempts to access o2 in C – This is allowed, s2 can read in C because it is s2's clearance level

The fact that the file is still there when s2 tries to access it can leak one bit of information (e.g., s1 tells s2 that when the troops will start moving he will make o2 disappear)

Covert channels

COVERT CHANNEL

any channel that allows information flows contrary to the security policy

Storage channels

e.g. shared counters, ID fields, file meta-data, etc.

Timing channels

e.g. use of CPU, load to memory (cache), queuing time, etc.

There are many possible **covert channels** in a system. These channels are enabled by **shared resources** that can be accessed by subjects with different clearance levels.

This reminds us that violating the principle of Last Common Mechanism is opening the door to problems.

Covert channels

COVERT CHANNEL

any channel that allows information flows contrary to the security policy

Storage channels

e.g. shared counters, ID fields, file meta-data, etc.

Timing channels

e.g. use of CPU, load to memory (cache), queuing time, etc.

Principle 7

Least common mechanism

The more resources are shared, the harder it is to eliminate covert channels

25

There are many possible **covert channels** in a system. These channels are enabled by **shared resources** that can be accessed by subjects with different clearance levels.

This reminds us that violating the principle of Last Common Mechanism is opening the door to problems.

Mitigating Covert channels

COVERT CHANNEL

any channel that allows information flows contrary to the security policy

Mitigation: isolation (communication with low level not possible) or add of noise to communication.

- Hard to achieve less than 1 bit / sec
- OK for documents, **NOT** OK for cryptographic keys
  - DoD policy: cryptographic keys must always be stored on dedicated hardware.

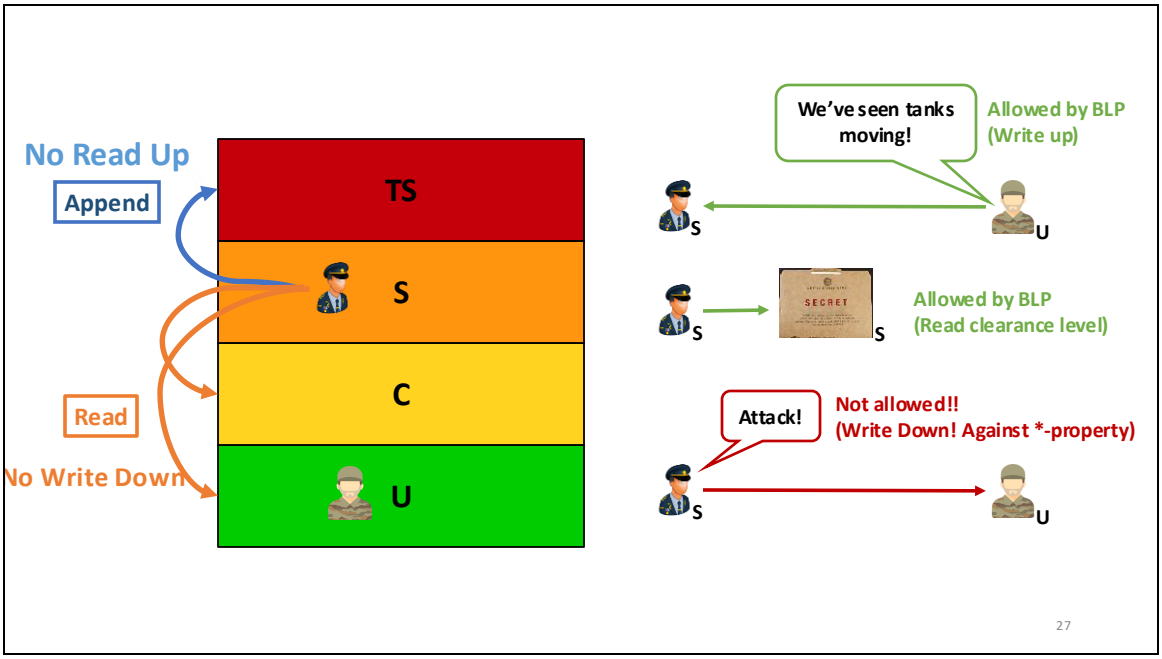
26

To eliminate covert channels one can:

- Isolate the high level: once information is up, do not let it go down (or limit this channel to minimum capacity). This however, it is not practical as working without acknowledgemets is very hard
- Add noise to communication: add fake interaction between high and low levels so that real intended covert channels cannot be distinguished from random noise.

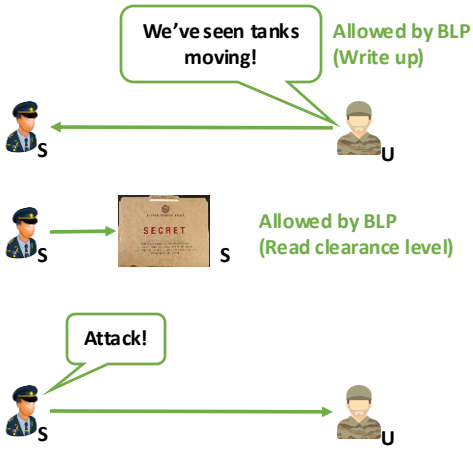
None of this option completely eliminate covert channels, it just slows them down a lot. But even 1 bit/second, which seems like a lot if we are trying to leak a 3Mb document (more than 1000h to download), is not much when we consider cryptographic keys (512 bits for symmetric cryptography – 8 minutes; or 2048 for asymmetric – 34 minutes).

This is why cryptographic keys are typically stored on dedicated secure hardware with very limited APIs.



To illustrate that not allowing level changes is problematic think of the example in the slide. If there is NWD, how will the General communicate with the soldiers after he has seen any secret document?

Declassification



DECLASSIFICATION

remove classification label

It is very typical and necessary

Under the control of the security policy.

- It **cannot** be made inherently safe

(*manual process*)

-Rules about archives, historical records

Hard to rule out covert channels.

How to know the object does not contain secrets?

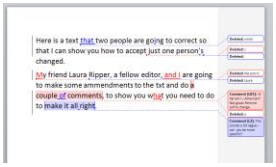
28

To enable communication with lower layers, which is essential and very normal in computer systems, one needs to include a **Declassification** process. Declassification is the process by which an object at a higher level is cleaned from information so that it can be passed down to lower levels.

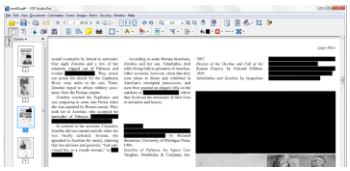
It must be noted, however, that it is very hard to rule out any information leakage, whether it is intentional (via a covert channel) or non intentional (see next slide)

Difficulty of Declassification in practice

- Microsoft Word revision history retains deleted text



- Portable Document Format (PDF) redaction by overlaying graphical elements (usually black rectangles) → the text is on the file!



Strategic adversary!

https://www.slideshare.net/ange4771/pdf-secrets

Hill, S., Zhou, Z., Saul, L., & Shacham, H. On the (in) effectiveness of mosaicing and blurring as tools for document redaction. *Proceedings on Privacy Enhancing Technologies*, *2016*.

Bell La Padula: limitations

- Confidentiality-oriented: does not consider integrity or availability
- State-based + single transition model: too low-level, not expressive
- The 3 security properties are not sufficient to ensure confidentiality…
  - Changes in clearance and classification can create covert channels
  - A static system without changes is impractical

BLP has three main problems:

- It only cares about confidentiality. Although the ds-property can be used to deal with integrity, BLP itself does not provide any integrity or availability protection
- The transitions enabled by the model are very low level to represent practical scenarios (one may need many transitions to represent a common operation); and the model is not expressive to deal with other security properties
- The three properties still do **not** guarantee that confidentiality is achieved

Computer Security (COM-301)

Mandatory Access Control
Integrity Security models

Carmela Troncoso
SPRING Lab
carmela.troncoso@epfl.ch

Some slides/ideas adapted from: George Danezis

Protecting integrity

Bell-La Padula focuses on **confidentiality**. Relevant for military / government environments

What about **commercial services?**

Banking, Stock and sales inventory, stock exchange, land registry, student grades database, electronic contracts, payments, …

Preventing fraud is about **protecting integrity**: the adversary has not influenced the result

Confidentiality is either *secondary* or *unnecessary*.

Integrity is key for computer security in general!!

TCB has to have high integrity.

Public key cryptography requires high-integrity for confidentiality

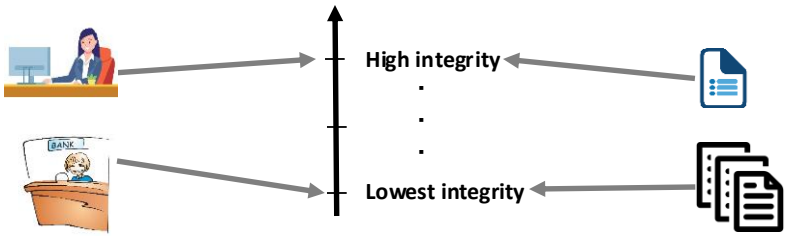
32

In the previous block we talked about Bell-La Padula. The main goal of this model is to ensure confidentiality of documents. Confidentiality is relevant in certain environments where secrecy is the top priority. We can think of military of government entities for which it is more important that the documents they hold are not read by others that are not modified.

In other environments, and in particular, for many entities with which we interact in our daily lives, confidentiality is not the first priority. For banks, registries, shops, confindentiality is important but not the main concern. Here, the main concern is the integrity of the accounts. It is indeed bad press if your accounting is made public, but what is really crucial is that your accounting is correct and has not been modified.

The need for integrity is not only important for commercial and economic activities. It is in fact **key** for computer security. Recall that the security of the system relies on the TCB. Therefore, the integrity of the TCB needs to be maintained!
Another example is Public key cryptography. As we will see in the applied cryptography lecture, if the encryption keys are corrupted during exchange one cannot guarantee anymore that only the intended receiver can see the information.

The BIBA model for integrity



Two operations:
"read" (observe)
"write" (modify)

Two key rules (strict)

- ***simple integrity (no read-down)***: protects higher integrity principals from being corrupted by lower integrity level data

- ****-integrity (no write-up)***: prevents lower integrity principals from corrupting high integrity data

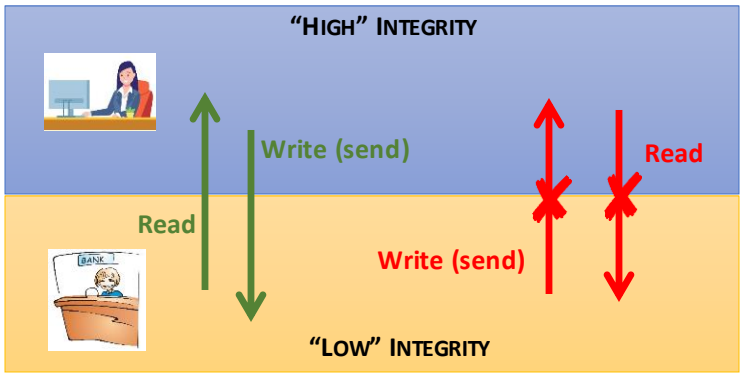
33

Similar as in BLP, in Biba subjects and objetcs are assigned an **integrity level**

There are two operations (as opposed to four). For instance, executing is not considered.

The two key properties are the dual of BLP:
- high level subjects should never see information of lower levels, so as not corrupt their vision
- for the same reason, low level subjects should never write in high levels so that they cannot pollute the high-integrity information

BIBA illustrated



EXAMPLES

In the Bank:
Director can establish a rule and every employee reads. Employees cannot rewrite rules

In the computer:
Web application open in the browser should not write to the file system (at most /tmp)

34

Summary of the Biba model:

High integrity subjects **can** write down, but **cannot** read from down

Low integrity subjects **can** read from up, but **cannot** write up

BIBA variant 1: Low-water-mark for subjects

Low-water-mark policy for subjects

- Subjects start processes at their highest integrity level.
- When accessing an object, its current level is lowered to the lowest of the two: current-level(s) and level(o)

Temporary downgrade for the session

- Example: mitigate impact of a network Trojan
- Hard to avoid label creep

35

If a subject (e.g., a process) needs to interact with an object in a lower level, then its current level is downgraded to low.

You can see this as a similar operation as “sandboxing”. Execute the process with dirty information in a safe environment.

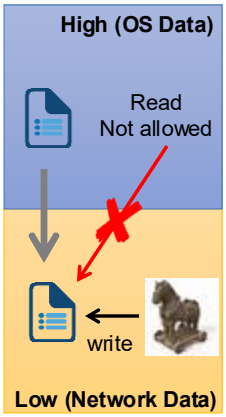
A problem with this process is that you could have a label creep problem in which all the subjects would end up in a lower level, or one subject is continuously lowered, and no-one can write in a higher level anymore

BIBA variant 2: Low-water-mark for objects

Low-water-mark policy for objects

- Once an object has been written to by a subject, it assumed the lowest level of the object or subject.

A high-integrity database written to by a process with access to the network (low integrity) is labelled at "low" integrity



What is the effect?

Dangerous! only allows for integrity violation detection

Mitigation: replicate objects & sanitize / erase

If a subject "pollutes" an object with information (by writing on it) this object is automatically downgraded to the level of the subject to avoid problems.

This policy does only allow for integrity violation detection. The file cannot harm, but the tampering has happened.

An option to avoid prevent the tampering is, instead of downgrading the real object, replicate it. After the operation has happened either sanitize and upgrade (see slide below) or just delete the replica.

BIBA Additional actions: Invoke

Simple Invocation

Only allow subjects to invoke subjects with a label they dominate

+ protect high integrity data from misuse by low integrity principals

- what level is the output?

Controlled Invocation

Only allow subjects to invoke subjects that dominate them

+ prevents corruption of high integrity data

- hard to detect polluting information

37

To avoid limitations, BIBA also allows the action invoke, so that subjects can interact with objects at other levels, but still protecting the object:

Here the Director acts as protection (can run checks or deny operations) for the object that the Teller needs to write on.

Invocation can be in one of the two directions:

- **Simple invocation:** High level invoke low level, hence protecting high level data. This has the problem that it is unclear which level the modified object has: low as the principal that performed the action, or high as the action was requested by a privileged principal?

- **Controlled invocation**: Low level invokes high level. The high level can act as protection, and in principle act as a warden to prevent the corruption of high integrity data but… we know it is hard to check that no secret/bad information is actually being written to the high integrity objects.

Sanitization

SANITIZATION

Process of taking objects with "low" integrity and "lifting them" to "high integrity"

"Sanitization" problems are the root cause of large classes of real-world security vulnerabilities

Malformed "low" (user) input can influence "high" (service) data and code

EXAMPLES

Web security: web server (high) accepts input from web client (low)
→ SQL interpreter → SQL injection vulnerability

OS Security: UNIX suid program (high) accepts input from a user (low)
→ short buffer → buffer overflow

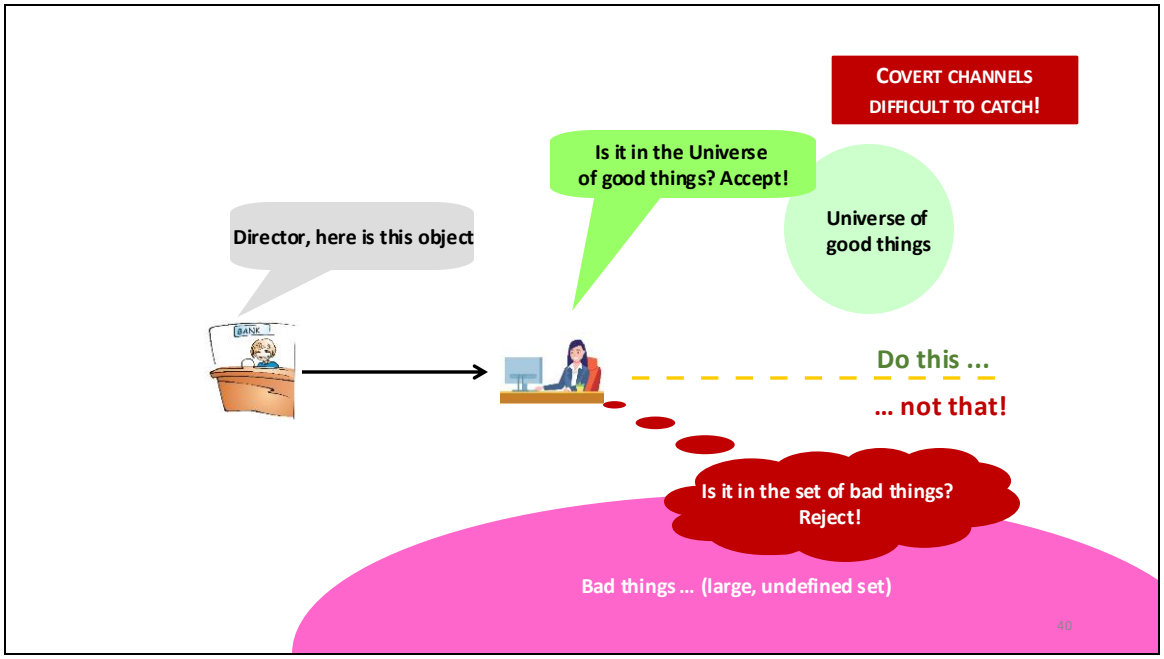
Fundamental principle of sanitization

PRINCIPLE 2: FAIL-SAFE DEFAULT
"Base access decisions on permission rather than exclusion" **[SS75]**

Positively verify that "low" objects are within a valid set before elevating their integrity to "high".

- White list: check that all properties of good objects hold.
- Do not blacklist: do not just check for bad objects or properties.

Insert a photo in a web album? Ensure caption is from a restricted set of Unicode, or apply to it a transform to "escape" / "encode" any characters not from that safe set into it. Do not simply check it does not contain "<script>". (XSS Attack)



Remember that we cannot know the full extent of the Universe of bad things. Therefore, checking whether an input is in that universe in order to reject is not a good practice. Your blacklist is never guaranteed to be complete.

The good practice is to check whether the inputs are part of the Universe of good things, which is actually well defined and much smaller.

Even if an input complies with the rules of the good things, it does not mean that there is no covert channel. As we know, they hare very hard to avoid.

Principles to support integrity

Three principles to guide your choices:

- **SEPARATION OF DUTIES**: Require multiple principals to perform an operation

(harder for an adversary to tamper with the system as they have to corrupt two principals)

- **ROTATION OF DUTIES**: Allow a principal only a limited time on any particular role and limit other actions while in this role

(harder for an adversarial insider to tamper with the system)

- **SECURE LOGGING**: Tamper evident log to recover from integrity failures. Consistency of log across multiple entities is key.

(harder to make an integrity breach durable)

41

Three principles that help designing and implementing security mechanisms that provide strong integrity.

- The principle of **separation of duties** is also known as "segregation" or "dual control". It implements the "separation of privilege" security principle. Why is this principle a good way to ensure integrity? when more than one entity is involved in an operation it is harder for one single entity to tamper with the integrity of the system and its outputs.

Examples:

- Accountant records payments / income, and stores checks deliveries/sales. The two must match!

- Both sides of a transaction need to keep a record, and the records must match (legal receipts)

- Two officers are required to launch a missile

- Developers should not also be operators

The principle of **rotation of duties** limits the time a principal is responsible for an action. The least time a principal is responsible for an action, the least opportunities this principal has to tamper with this action. This also implements in a way the separation of privilege principle, in the sense that for each action there are more than one principal responsible over time. In addition, as new principals become responsible, they act as auditors of the previous principals increasing the chance of detecting misbehaviour.

Examples:

- *Guards appointed for a single (random) shift to guard the bank safe*
- *Tellers changed over time so that they cannot fiddle with accounts*

An final principle to increase the chance of detecting misbehaviour, and recover from integrity problems, is the use of logging. This in fact implements the principle of "Compromise recording".

Secure logging means that the log itself is tamper resistant: i.e., a principal that can tamper with the system integrity cannot change the value of the log to hide her actions.
To enable secure logging it is important to consider the Separation and Rotation of Duties principles.

Examples:

- *Log of transactions inside the ATM machine.*
- *Cash till has an internal log.*
- *Notaries maintain logs of transactions and contracts.*
- *Bitcoin!*

Chinese Wall model

Inspiration: UK rules about handling "conflicts of interest" in the financial sector.

- A separation must exist at all times, even within the same firm, between people engaging in activities that conflict with each other.
- Cost of failure: large fines and reputation

David FC. Brewer and Michael J. Nash. "The Chinese Wall Security Policy." in IEEE SSP 1989

Chinese Wall model: Entities and Basic Concepts

1. All objects are associated with a label denoting their origin

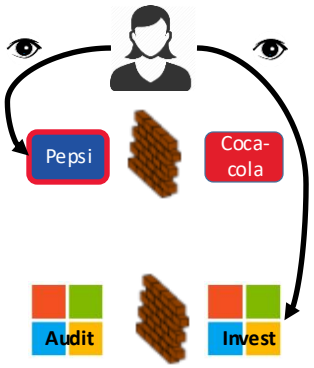
   "Pepsi Ltd.", "Coca-Cola Co.", "Microsoft Audit", "Microsoft Investments"

2. The originators define "conflict sets" of labels

   {"Pepsi Ltd.", "Coca-Cola Co."}, {"Microsoft Audit", "Microsoft Investments"}

3. Subjects are associated with a history of their accesses to objects, and in particular their labels.

ALICE
1. *Pepsi*
2. *Microsoft Invest*

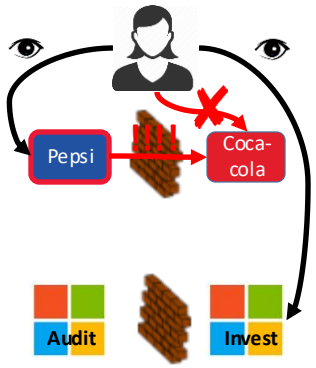


Chinese Wall model: Access rules

A subject can read an object (for either read or write) if the access ***does not allow an information flow*** between items with labels in the same conflict set

Alice starts her first day at work

1) She accesses files of "Pepsi Ltd" (OK)
2) She accesses files of "Microsoft invest" (OK)
3) She tries to access files of "Coca-cola Co." (access denied!)



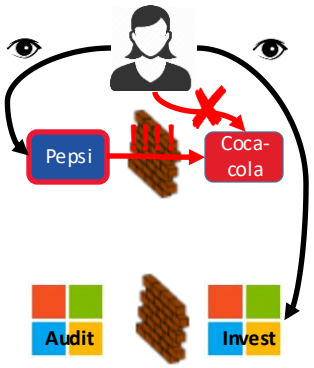
Chinese Wall model: Access rules

A subject can read an object (for either read or write) if the access ***does not allow an information flow*** between items with labels in the same conflict set

Alice starts her first day at work

1) She accesses files of "Pepsi Ltd" (OK)
2) She accesses files of "Microsoft invest" (OK)
3) She tries to access files of "Coca-cola Co." (access denied!)

Why? She has already accessed files from "Pepsi Ltd" thus an information flow between those and "Coca-cola Co" might happen (She could work again with "Pepsi")

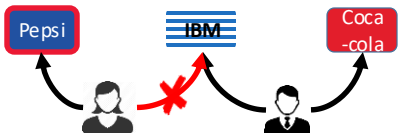


Chinese Wall model: Indirect flows

Direct flow within a conflict set is easy to detect! What about indirect?

Alice and Bob start together

1) Alice is assigned to “Pepsi Ltd” (OK)
2) Bob is assigned to “Coca-cola Co.” and “IBM Co.” (OK)
3) Alice tries to access files of “IBM Co.” (access denied!)



As in BIBA, one can also have sanitization to enable flexibility in Chinese Wall models. While some information learned from Pepsi should never get to Coca-cola via IBM, one can sanitize (remember, this is hard) some of the inputs to IBM so that Alice can actually work with more clients!

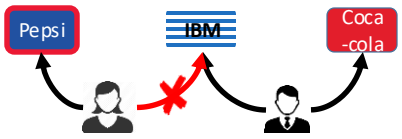
Chinese Wall model: Indirect flows

Direct flow within a conflict set is easy to detect! What about indirect?

Alice and Bob start together

1) Alice is assigned to “Pepsi Ltd” (OK)
2) Bob is assigned to “Coca-cola Co.” and “IBM Co.” (OK)
3) Alice tries to access files of “IBM Co.” (access denied!)

Why? If she writes in IBM with her knowledge of Pepsi, then the information ***may*** flow to Coca-cola.



As in BIBA, one can also have sanitization to enable flexibility in Chinese Wall models. While some information learned from Pepsi should never get to Coca-cola via IBM, one can sanitize (remember, this is hard) some of the inputs to IBM so that Alice can actually work with more clients!

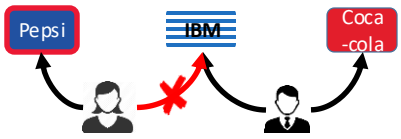
Chinese Wall model: Indirect flows

Direct flow within a conflict set is easy to detect! What about indirect?

Alice and Bob start together

1) Alice is assigned to "Pepsi Ltd" (OK)
2) Bob is assigned to "Coca-cola Co." and "IBM Co." (OK)
3) Alice tries to access files of "IBM Co." (access denied!)

Why? If she writes in IBM with her knowledge of Pepsi, then the information ***may*** flow to Coca-cola.



SANITIZATION is necessary for business
"Un-label" some items as long as the information cannot lead to any conflict of interest, e.g., extract some "general market information"

48

As in BIBA, one can also have sanitization to enable flexibility in Chinese Wall models. While some information learned from Pepsi should never get to Coca-cola via IBM, one can sanitize (remember, this is hard) some of the inputs to IBM so that Alice can actually work with more clients!

Summary of the lecture

- **Security models**: patterns to design MAC policies

- **BLP**: Confidentiality
  - Key concept: Declassification
- **BIBA**: Integrity
  - Can bootstrap: high confidentiality (PKI) or High availability (replication)
  - Can lead to: low confidentiality or low availability
  - Key concept: Sanitization
- **Chinese Wall**: Conflicts of interest (confidentiality & integrity)
- **Multilateral security**: conflicting properties

Biba focuses on integrity, but having integrity helps bootstrapping systems that have other properties:

- High integrity of public keys is essential to create systems that enable high confidentiality, such as in Public Key Infrastructure (see Applied cryptography and Network Security lectures)

- High integrity of logs is essential for replication to be useful and provide high availability.