Computer Security

Carmela Troncoso, KU Leuven (COSIC)

Computer Security Course, University of Vigo

21st-July-2009

Acknowledgements: Dr. George Danezis

Outline

- Introduction
- Access control matrix
  - Access Control Lists
  - Capabilities
- Discretionary access
- Mandatory access: Security policies
  - Secrecy: Bell-LaPadula
  - Integrity: Biba
- Implementation is not trivial
- Certification
- Conclusions

“*Is the discipline that deals with the prevention and detection of unauthorised actions by users of a computer system*”

D. Gollmann, Computer Security (1999)

History

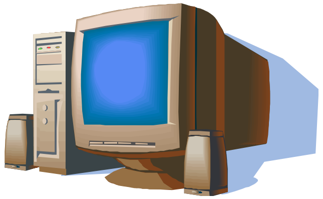
- 50-60s: Mainframe computer
  - punch cards, paper tape, and/or magnetic tape
  - No interaction, batch processes

IBM650 (1954)

- 60-70s: terminals connected to the mainframe
  - Several users on one computer
  - One domain administrator
    - Security = no interferences
    - Permissions and access control

Televideo925

- 70-80s: PC
  - One user one computer
  - “No need” for security: user presence

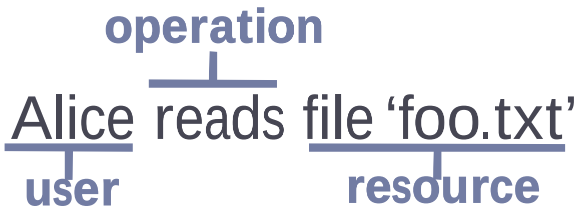


Nowadays

- Networked PCs
  - and servers, databases, clouds,…
  - Untrusted content
  - Untrusted code running
  - …
- Back to one computer many users
  - No physical security
- Network services
  - Back to security problems! and worse than before…

Key Concepts

- Main actors
  - Principals/users (subject)
  - Resources (object)
  - Operations (action: read, write, append, execute...)



  - Alice reads file ‘foo.txt’

- Only **authorized** principals should perform **authorized** operations on **authorized** resources

Not that easy...

- What/who are the principals?
  - User = physical person or process?
  - Accountability: are users responsible for their programs?
  - Intentionality: what if there is a bug?
- Granularity of the resources:
  - Hardware: actual processors
  - Kernel: memory pages
  - OS: files, sockets
  - Application: DB records, user accounts
- Where do we implement security?
  - Do not build a castle in the sand...

Not that easy...

- What should be protected?
  - Data/resource (number = integer)
  - Operations (open account only by bank clerks)
  - Users (who can access the data)

- Access Control Matrix

	foo1.txt	foo2.txt	foo3.txt
Alice	write	read, execute	-
Bob	-	read,write	-
Charlie	execute	-	read,write,execute

- Which is the best way to store it?

Capabilities

- Capabilities (by row): *principal-oriented*
  - *Alice: foo1.txt → write, foo2.txt→read, execute*
  - *Bob: foo2.txt → read, write*
  - *Charlie: foo1.txt → execute, foo3.txt→read,write,execute*
  - Who has rights on *foo1.txt*? Which ones?
  - Runtime checking is fast
  - Delegation is easy
    - Delegated capabilities revokation is difficult

Access Control Lists

- Access Control Lists (by column): *object-oriented*
  - *foo1.txt: Alice → write, Charlie → execute*
  - *foo2.txt : Alice → execute, Bob → read, write*
  - *foo3.txt : Charlie → read, write, execute*

  - Revokation not trivial (e.g., a user leaves the system)

and there is more…

- Privileges
  - Principals can be temporarily granted rights
  - Administration tasks
- Groups
  - Simplify access control policy
  - Aggregates users with similar rights
  - Permission to the whole group
- Deletion, ownership,…

Who sets the Access Control Matrix?

- Discretionary Access Control
  - Users set permissions
  - Ownership of resources (UNIX, Windows)
  - Users in charge of their security

- Mandatory Access Control
  - Security policy set by “authority”
  - Hard security constraints:
    - Medical environments (confidentiality, integrity)
    - Military (Confidentiality)
    - Banking (Integrity)

Discretionary Access: UNIX

- Entities
  - All resources are files (files, devices, sockets,...)
  - Files belong to a user and group
  - read/write/execute granted to user/group/world ~ RBAC
- Users set permissions
  - Stored in `iNodes` = Access Control Lists
- Superuser `root`

UNIX security problems

- Who is the principal?
  - Executables run with the rights of the user executing them!
- Shared resources?
  - Example: `sendmail`
    - All received emails in the same file
    - Users only access their emails, cannot grant read to them
- Privileges: `suid-bit`
  - Executables run as their owner, not the executing user
  - `sendmail` reads file and selects users' emails
- Problem!

Mandatory access: Security policies

- The access control matrix implements a security policy
  - Sets which assets to protect and how – high level
  - Complex, high level risk management
  - Appropriate strength of security mechanisms
  - Security policy is analogous to Law

- But given a set of constraints is undecidable if a matrix satisfies them…
- … we can never decide if an access control system is safe! [Harrison-Ruzzo-Ullman]

Example

Who has access to the key of the room?

Easy: keys are only given to the professor that reserved the room

but... he may want to send somebody else to reception: student temporarily granted "professor rights"...

the student may make a copy...! or lose the key!

also... emergency situations key is given without reservation

and... what about the cleaning staff that has access to the full building?

What is a policy?

- A **security policy** is a statement that partitions the system into a set of authorised (secure) states and a set of unauthorised (nonsecure) states
  - User actions make the system transition from one state to another

- A **secure system** is a system that starts in an authorised state and cannot enter an unauthorised state.

- A **breach of security** occurs when a system enters an unauthorised state.
  - Need to define carefully (e.g, copying homework)

Types of policies

- **Confidentiality policy** :
  - Information leakage to anoutharized entities
  - Leakage of rights
  - Information flow without leakage of rights

- **Integrity policy:**
  - Which ways information may be altered.
  - Which entities can alter it.

Access Control Policy Models

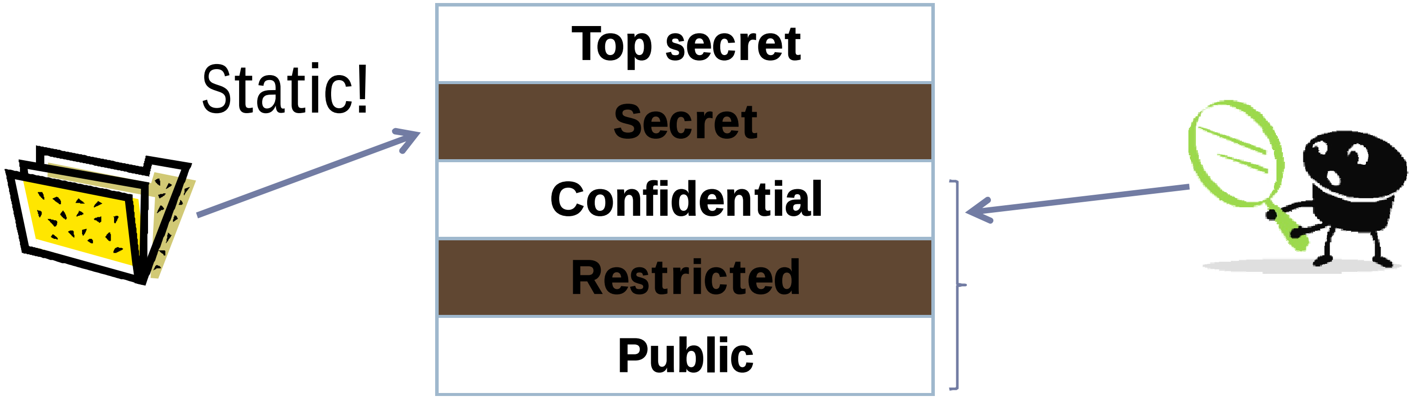
- Set patterns to ease the process: *Security labels* for objects (sensitivity), with *security clearances* for subjects (authorization).

- Formal representation proved to fulfill certain properties
  - Confidentiality,
  - Integrity,
  - Separation of duties, ...

- Not everything is solved...
  - Who manages the policy?
  - Policies need to be adapted
  - Only safe case

Bell-LaPadula model (BLP)

- Ensures **Confidentiality**
  - Developed as part of U.S. government funded research at the MITRE corporation on security models and the prevention of disclosure threats in multi-user operating systems.
  - Basis of several standards, including DoD's Trusted Computer System Evaluation Criteria ("Orange Book").



Static!

Top secret

Secret

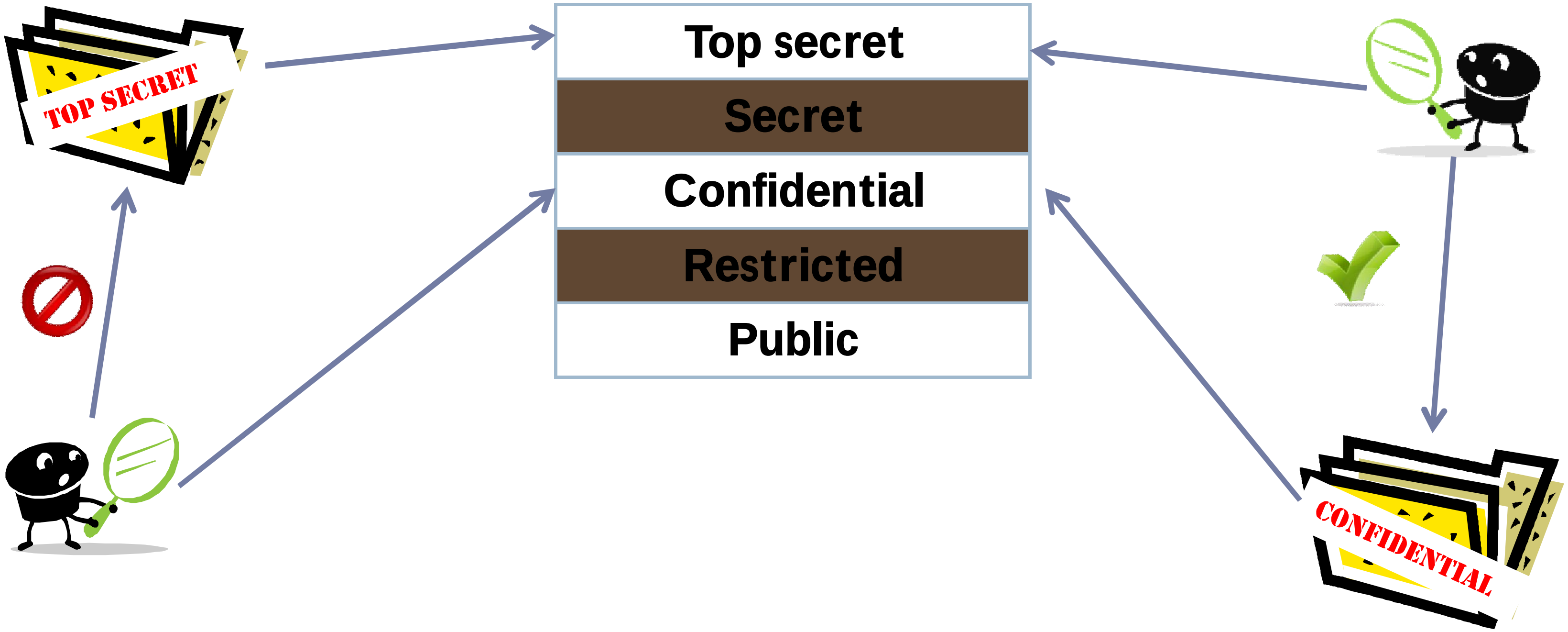
Confidential

Restricted

Public

BLP Rules: no-read-up (NRU)

- Simple security property (ss-property)
- Unauthorized subjects cannot see sensitive objects



Top secret

Secret

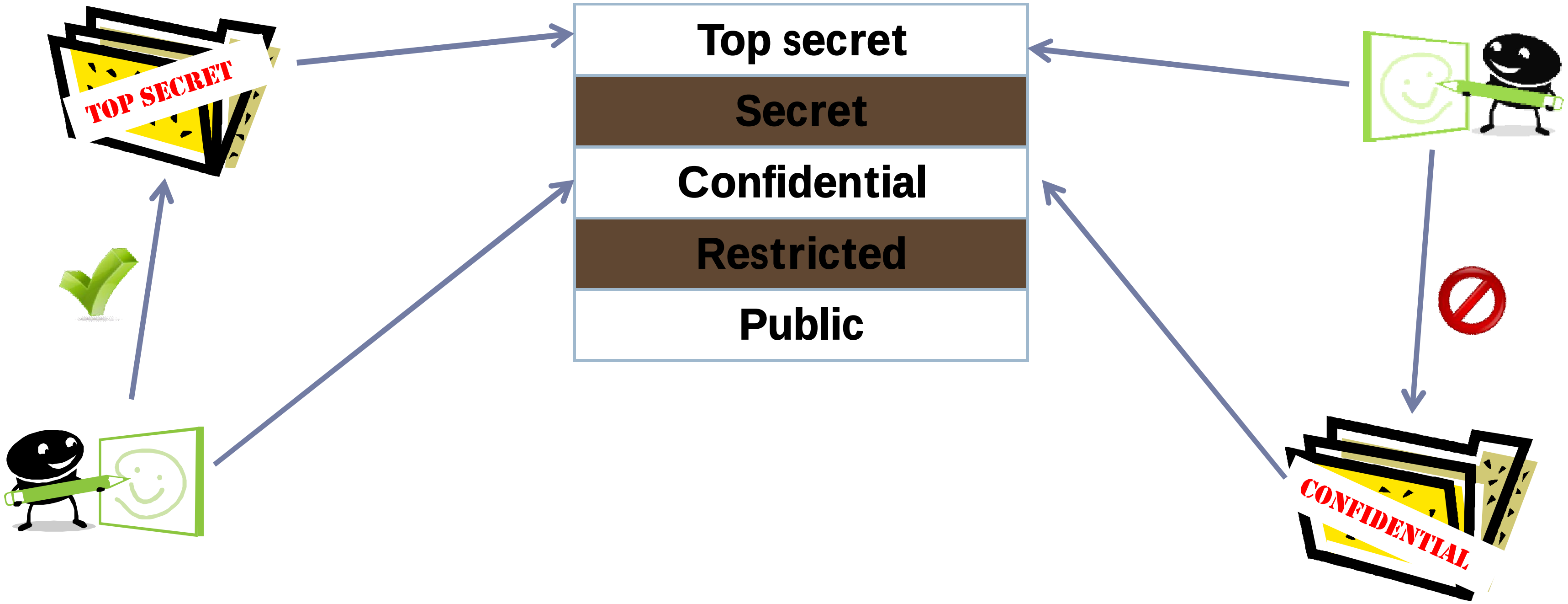
Confidential

Restricted

Public

BLP Rules: no-write-down (NWD)

- Star property (*-property)
- Trusted subjects cannot write unclassified objects



Top secret

Secret

Confidential

Restricted

Public

Limitations of BLP

- Static!
  - Tranquility property: users do not change labels in a way that the policy is violated
  - Not very useful… who changes the policy then?

- Existence of cover channels
  - Information flow not controlled by a security mechanism
  - Process at high signals process at low, denial of access
  - Exploitable by principals/malware (trojan horse scenario)
  - Shared resources leak information

Limitations of BLP

- Polyinstantiation
  - Different levels = different value
  - Hide or lie?

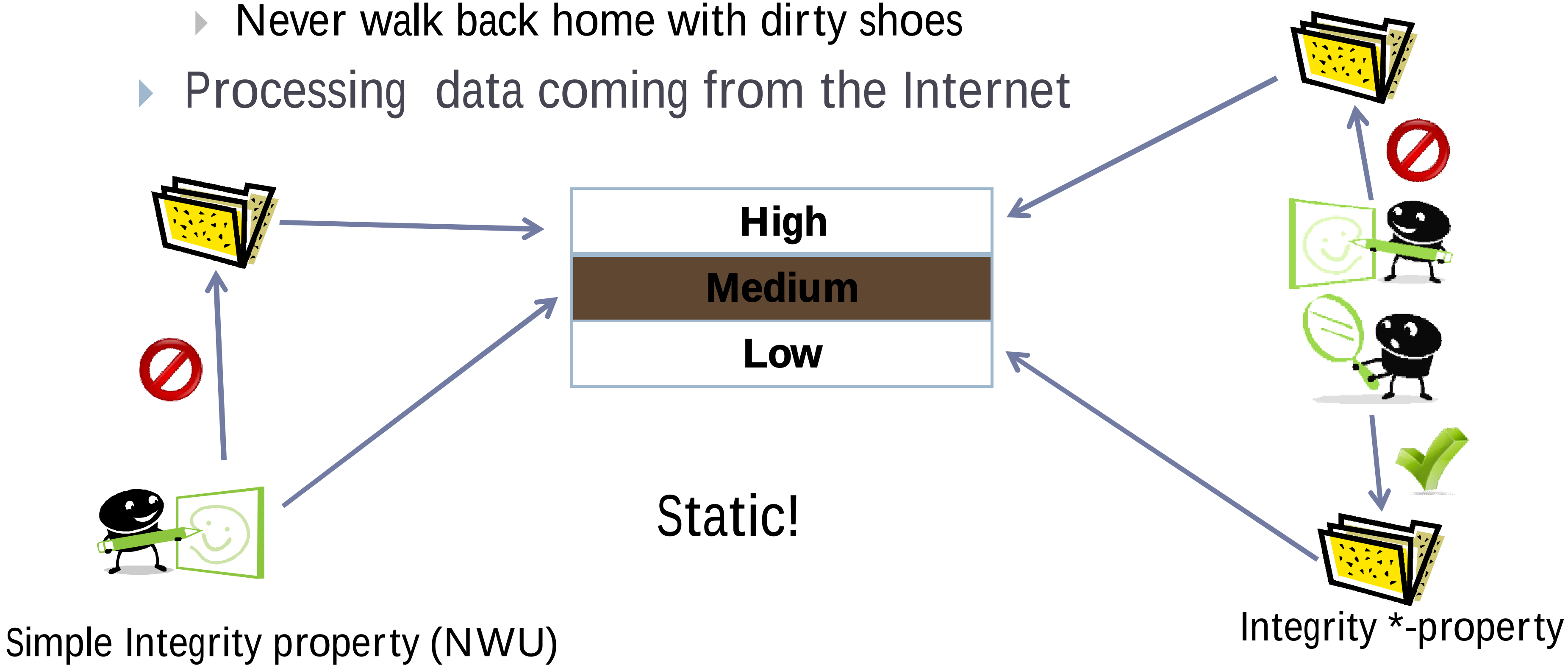
- Bloat at the top
  - Information only goes up
  - Need for *declassification*
    - *Solves the bloat…*
    - *…but introduces covert channels*
    - Job of declassification often not trivial
      - □ e.g., Microsoft word saves a lot of undo information

Implementations of BLP

- Air-gap security
  - Guards with guns & separate rooms for high and low
  - No media can go from high to low
- The NRL pump
  - One way network
  - Not easy: without acks
- Secure operating systems
  - Can only limit covert channels to (1 bit / second)
  - Ok for big secrets, not ok for keys (use hardware for those)

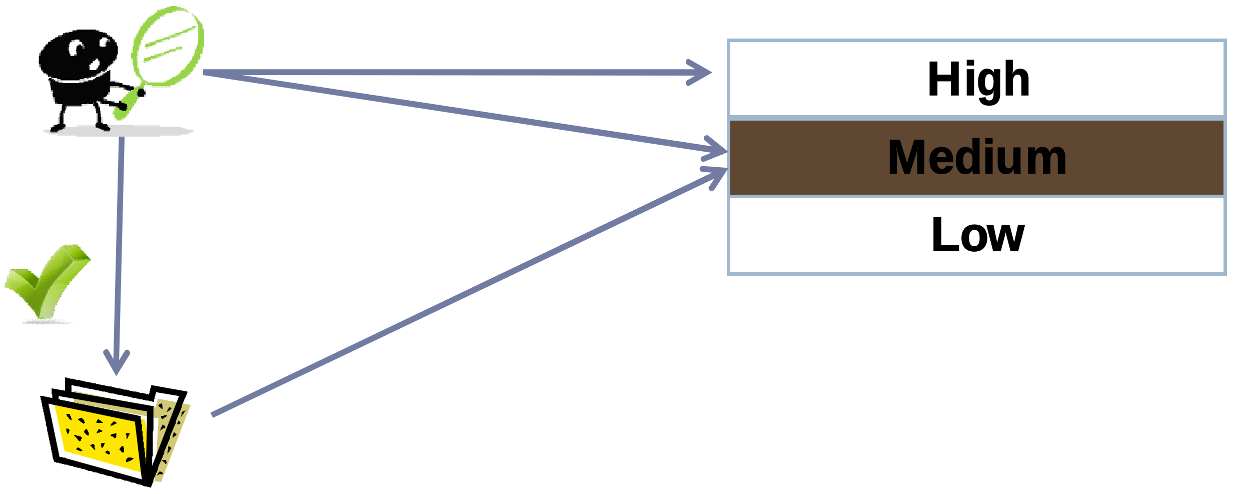
Biba model

- Ensures **Integrity**
  - NRU and NWD ensure confidentiality, but WU and RD introduce integrity problems
    - Never walk back home with dirty shoes
  - Processing data coming from the Internet



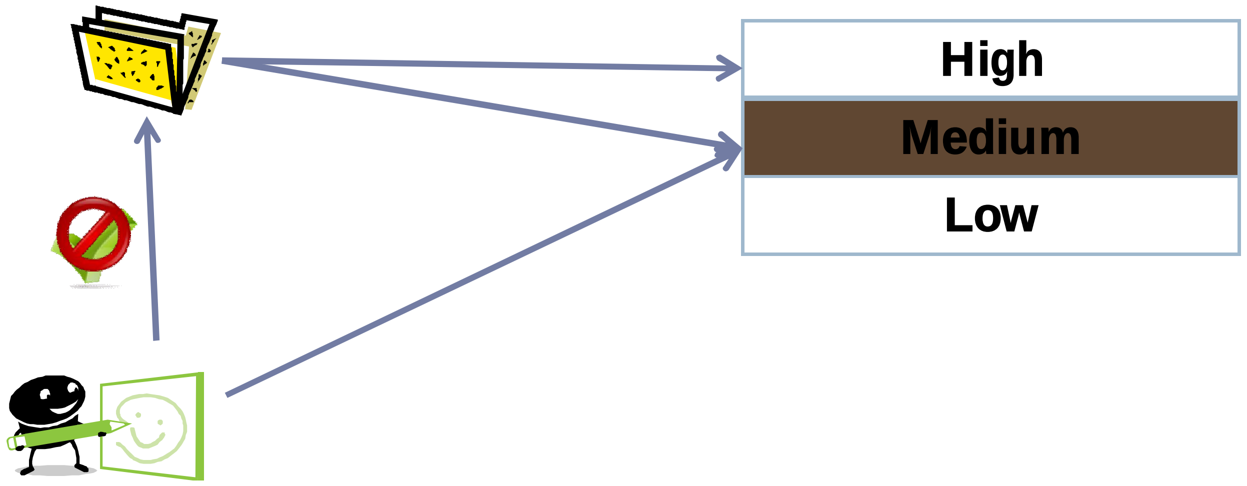
Biba Dynamic Integrity Levels

- Subject low watermark property
  - Allow a subject to read down, but first lower its integrity level to that of the object being read.



Biba Dynamic Integrity Levels

- Object low watermark property
  - Lower object level to that of subject doing the write.



Invocation policies

- Now the bloat is at the bottom
  - Need for *sanitization…*

- …or **Invocation**:
  - Invocation - subject can only invoke another subject at or below its own integrity level
  - Controlled Invocation – Low-level subjects should have access to high-level objects only through high-level tools
  - Ring Property – Subjects should not be allowed to use tools at integrity levels below their own

Biba model discussion

- Does not address data consistency
- Only prevention of modifications by unauthorized users
  - Authorized users can still make improper modifications
- Problem to assign appropriate integrity levels
  - What is integrity?
- Only implemented in few systems

Chinese-Wall model

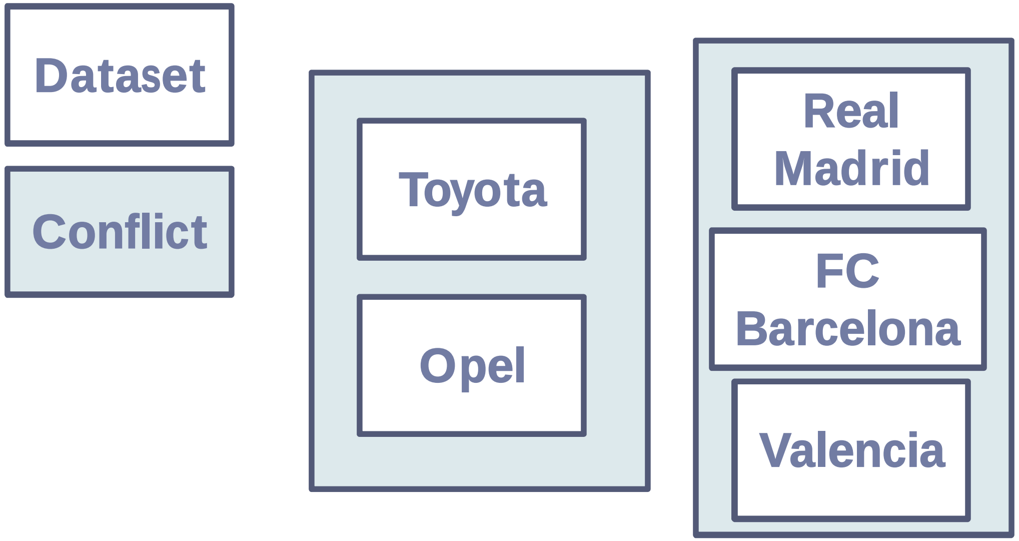
- Commercially inspired: no conflicts of interest should arise (Consultancy environment).

- Informally, conflicts arise
  - because clients are direct competitors, or
  - because of the ownership of companies.

- There **must not** exist an information flow that creates a conflict of interest

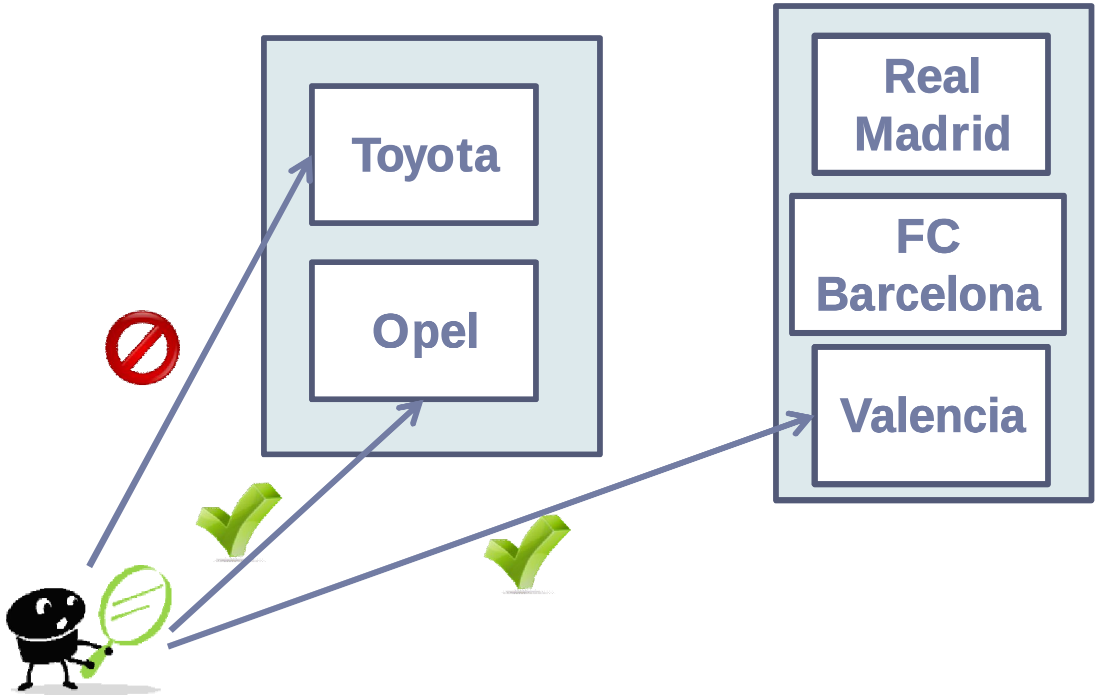
Chinese-Wall model

- Objects contain information from a single company
  - Grouped in Company Datasets
- Subjects have access to objects (consultancy analyst)
- Conflicts of interest: set of companies that should not learn about one object.



Chinese-Wall model

- A subject can access any information as long as it has never accessed information from a different company in the same conflict class.



- Permissions have to be checked **dynamically!**
- Cover channels still exist

Clark-Wilson model

- Data integrity and consistency control
  - Used by banks
  - Objects must be always in a consistent state

- Emphasis on integrity
  - internal consistency
  - external consistency

- Instead of (Data-Level) move to (Data-Transaction)

Clark Wilson Mechanisms for Integrity

- Well formed transactions
  - Only process data using *constrained transactions* that ensure data integrity (consistent states)
    - e.g., use a write-only log to record all transactions
    - e.g., double-entry bookkeeping
  - Security is reduced to integrity of transactions

- Separation of duties
  - Certifier: entity that certifies the correctness of a transaction
  - Certifier and the implementer be different entities.

Information-flow models

- Not only the direct flow through access operations modeled by BLP.
- Information-flow from an object x to an object y, if we may learn more about x by observing y.
  - If x=0 then y=1
- Undecidable!

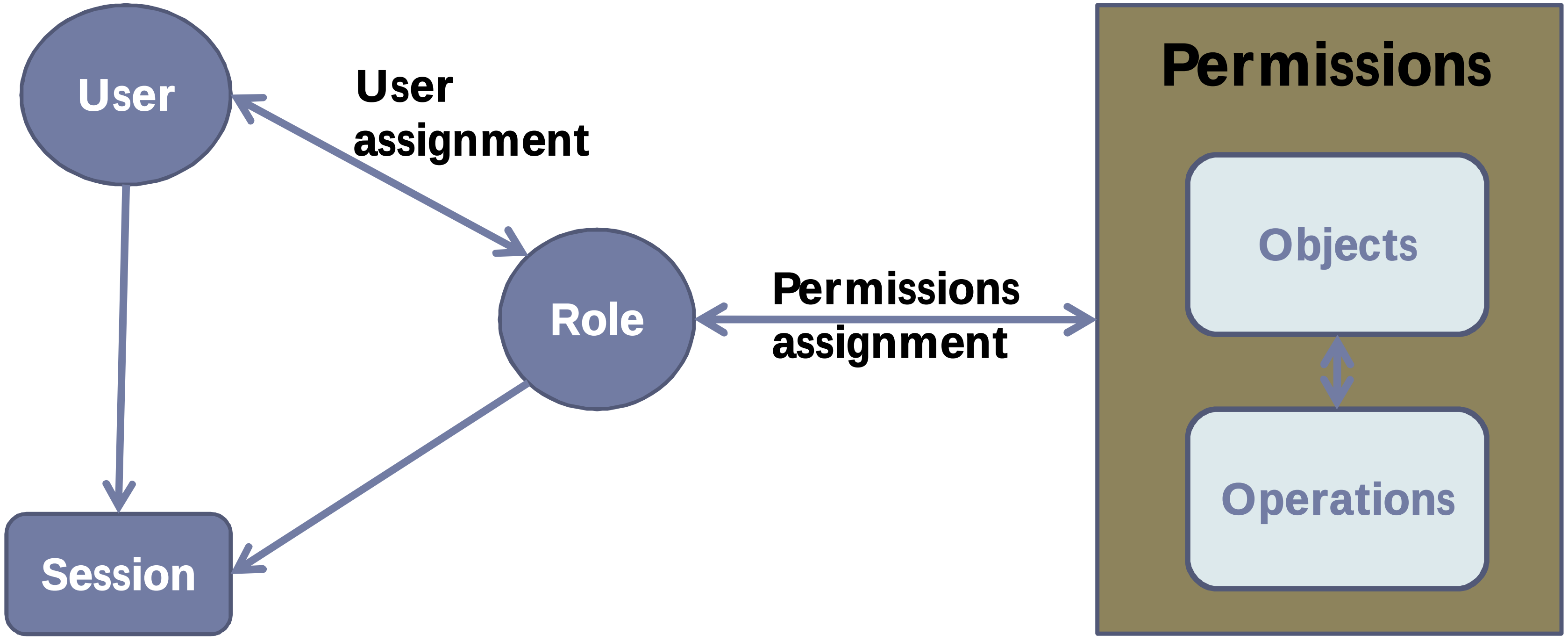
Role Based Acess Control (RBAC)

- A new level of indirection
  - Users associated to roles not to objects
  - Generalization of Clark-Wilson
- A Role is a set of procedures:
  - Concierge
  - Student
  - Professor
- Rights depend on the role being performed

Role Based Acess Control

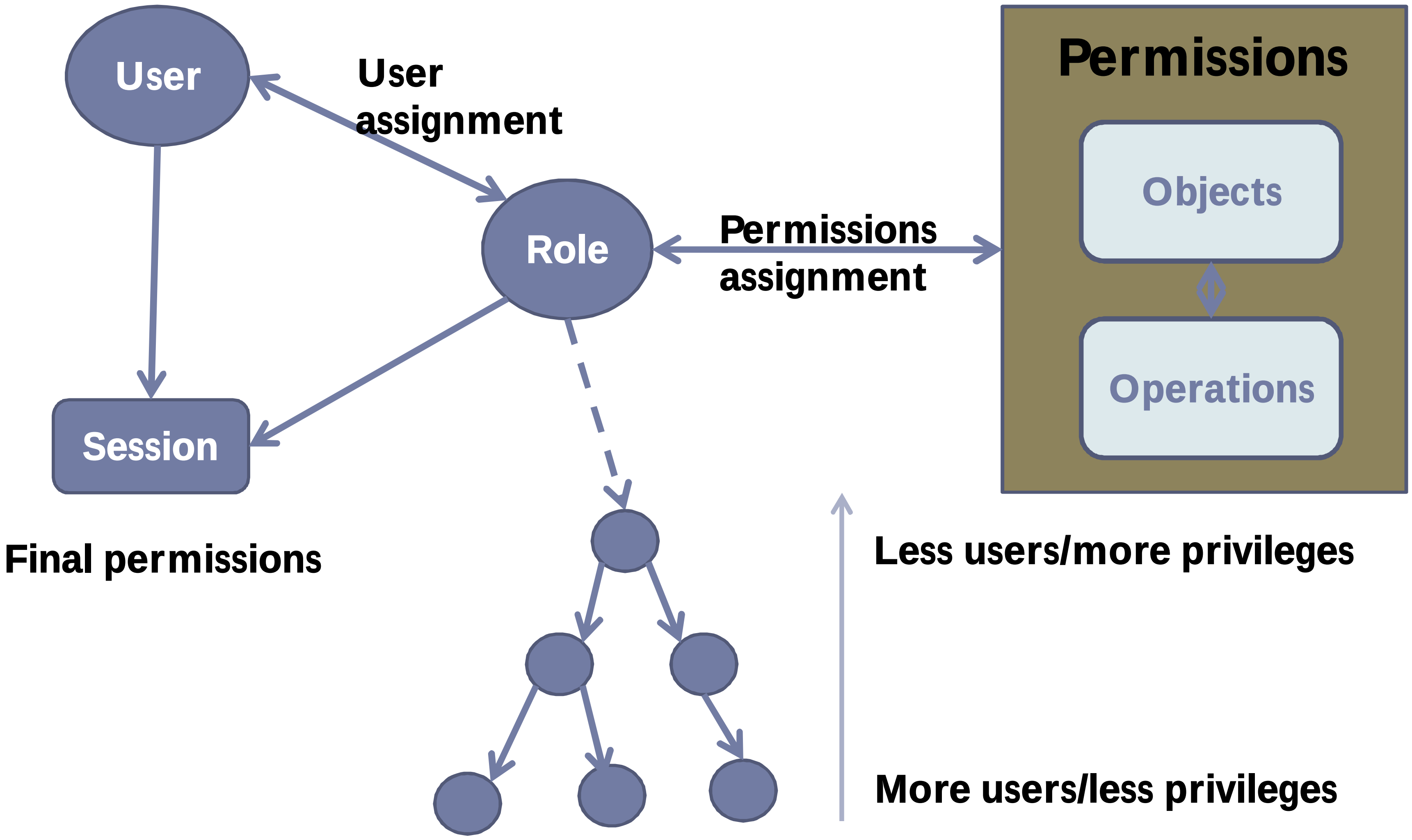
- **Least privilege principle**
  - Roles are allowed only the absolute necessary principles
- Memberships of users to roles do not change role privileges
- NIST reference models
  - Core RBAC
  - Hierarchical RBAC
  - Constraint RBAC
  - Consolidated RBAC (Hierarchical+Constrained)

Core RBAC



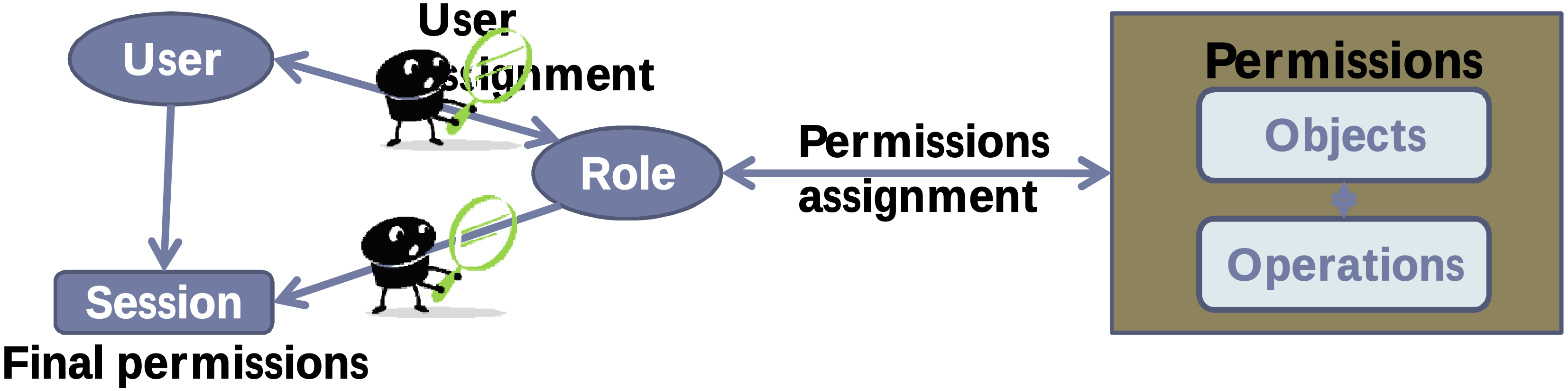
Final permissions

Hierarchical RBAC



Constrained RBAC

- Conflicts of interest
  - User having conflicting roles
  - Inheritance breaking conflicts of interest
- Separation of duties
  - Static: clear conflicts on user assignment to roles
  - Dynamic: check conflicts during session
    - No two superusers active simultaneously



Policy vs. Mechanism

- Policy defines the safe state
  - Does not actually enforce it…

- Laws do not impede crime…
  - but chains, doors, barriers, police, … do

- A mechanism is an entity or procedure that enforces some part of the security policy
  - Access controls
  - Output control

Implementation of a Policy model (or any other security policy)

- Physical security…
  - Air-gap implementation
- … or Concept of a Trusted Computing Base (TCB)
  - Every element of hardware or software on which your security policy relies to be enforced.
  - Do not care about faults outside it
- Important principle: make it as **small & simple** as possible
  - Makes verification and certification easier
  - Code review, documentation, automated proofs

(Not that good) Example: UNIX

- In a Unix workstation, the TCB includes at least:
  - the operating system **kernel including all its device drivers**
  - all processes that run with **root privileges**
  - all **program files owned by root with the set-user-**ID–bit set
  - all **libraries and development tools that were used** to build the above
  - the **CPU**
  - the mass storage devices and their **firmware**
  - the **file servers and the integrity of their network links**

- A security vulnerability in any of these could be used to bypass the entire Unix access control!

The Fundamental Dilemma

“Security-unaware users have specific security requirements but usually no security expertise”

- Need for **security evaluation**
  - Check whether a product delivers the advertised security
  - Rainbow series: orange, red, (light) pink,...
  - Common Criteria

- Risk Analysis
  - Security vs. Performance
  - Security vs. Cost

Evaluating system security

- A formal security evaluation requires
  - System's functional requirements
  - System's assurance requirements
  - A methodology to determine if the system meets these requirements
  - A measure of evaluation
    - Referred to as **a level of trust**

- A formal evaluation methodology
  - A technique to measure how the system meets the security requirements

Evaluation methods

- Products should be evaluated throught all their life cycle

- Obtain a certificate of **trustworthiness**

- Historical development
  - Many standards:
    - TESEC 1983-1999 (The Orange Book)
    - ITSEC 1991-2001
    - Federal criteria 1992
    - FIPS 140-1 of 1994 and FIPS-2 of 2001
    - The common criteria 1998- present
    - Other commercial efforts

Orange Book (1983)

- U.S. DoD
  - Trusted Computer System Evaluation Criteria (TCSEC)
  - Basic requirements for assessing the effectiveness of computer security controls built into a computer system

- Individual accountability regardless of policy must be enforced (Auditability)

- Categories: describe the trust an individual or organization places on the evaluated system
  - D — Minimal protection
  - C — Discretionary protection
  - B — Mandatory protection
  - A — Verified protection

Criticisms of Orange Book

- Mixes various levels of abstraction in a single document
  - Documentation, testing,…
- Does not address integrity of data
  - Military based
- Combines functionality and assurance in a single linear rating scale

Common Criteria (1999)

- Common Criteria for Information Technology Security Evaluation (International standard ISO/IEC 15408)

- Framework in which
  - users can *specify* their security requirements,
  - vendors can then *implement* and/or make claims about the security attributes
  - testing laboratories can *evaluate* the products to determine if they actually meet the claims. In other words

- Assures that these processes have been conducted in a rigorous and standard manner

Common criteria elements

- Target of evaluation (TOE)
- Protection profile (PP): security requirements for devices
  - e.g., bank tokens
- Security target (ST): different PPs
  - Vendor targets capabilities
- Security functional requirements (SFR): individual functions
  - e.g., type of authentication, encryption scheme

Common Criteria Categories

- Evaluation Assurance Levels (EAL): depth of the evaluation
  - EAL1: tester **reads documentation, performs some functionality tests**
  - EAL2: developer provides **test documentation and vulnerability analysis for review**
  - EAL3: developer uses **RCS, provides more test and design documentation**
  - EAL4: low-level **design docs, some TCB source code,** secure delivery, independent vul. analysis (state of the art for commercial products)
  - EAL5: **Formal security policy, semiformal high-level** design, full TCB source code, independentTesting
  - EAL6: Well-structured source code, **reference monitor for access control, intensive pen Testing**
  - EAL7: **Formal high-level design and correctness proof of** implementation

Other evaluation guides

- (Light) Pink Book (1993)
  - Covert Channel Analysis of Trusted Systems
- Red Book (1987)
  - Trusted Network Interpretation: extending the Orange Book to Networks
- Rest of the Rainbow Series…

Limitations

- Certification is a costly (money and time) process,
- Certification of documentation,
- Criteria are ambiguous,
- Re-evaluation of a certified product,
- Procedures are old,
- Certificates apply to an specific version and configuration, **and at the end there is no security guarantee!!**

Other topics

- Roles & role mining
- How to present policies?
- Digital rights management
  - Seen as a BLP confidentiality model
  - Standard problems!
- Trusted computing
  - High integrity model
- Shared environments
  - Security policies for on-line games (integrity)
  - Security policies for social networking sites (privacy)
  - Security policies for Web Browsers (same origin, etc)
- Distributed systems security: same but more complex!

Conclusions

- Ensure that “only *authorized* principals should perform *authorized* operations on *authorized* resources” **is not easy**

- Each system has its own requirements, that depend on the environment: **there is no perfect recipe for security**

- Even if there was... **translate into implementation is not trivial**
  - What about networks?? (tomorrow)

Further reading

- Books:
  - Dieter Gollman, “Computer Security”
  - Ross Anderson, “Security Engineering”
  - Matt Bishop, “Computer Security (Art and Science)”
- Articles:
  - Ross Anderson and Roger Needham, “Programming Satan’s Computer”
- Standards:
  - ISO 27799 (How to manage security and make policies)
  - The Rainbow series