University of BRISTOL

Systems Security

COMSM1500

bristol.ac.uk

STUDENTS: DON'T MISS OUT

THINK AHEAD AND MAKE SURE YOU CAN VOTE IN THE GENERAL ELECTION, 12 DEC 2019

GET REGISTERED TO VOTE BEFORE 26 NOVEMBER

You only need to register once, but may need to change your details if you have changed your name or moved house.

TO REGISTER QUICKLY HAVE YOUR NI NUMBER READY

Don't have your National Insurance number? Call **0800 141 2075** or visit **gov.uk/apply-national-insurance-number** ASAP.

UNSURE IF YOU'LL BE AT YOUR HOME OR TERM-TIME ADDRESS?

You can register at both, but can only vote once in the general election. Register at **gov.uk/register-to-vote** Remember, it is an offence to vote twice.

CAN'T GET TO YOUR POLLING STATION ON 12 DECEMBER?

Once registered, apply for a postal vote before:

- 5pm on 26 November if you live in England, Scotland or Wales
- 5pm on 21 November if you live in Northern Ireland

Register today | gov.uk/register-to-vote

YOUR VOTE MATTERS
DON'T LOSE IT

Feedback on the unit

- Blackboard
- Unit evaluation (left menu)
- Mid Unit Evaluation Survey (TB1)
- Read instructions and fill the form!

Access Control

bristol.ac.uk

Access Control

- Physical access control
  - Physical enforcement to access to areas
- Digital access control
  - Restrict access to resources and interactions

Subjects and Objects

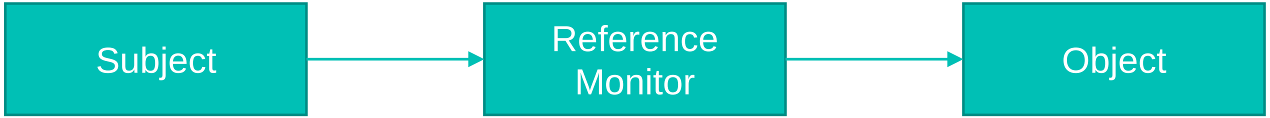
- Object
  - Passive entity that contains information
  - e.g. file, record, memory location etc.
- Access
  - Ability to perform an action on/interact with an object
  - Flow of information between a subject and an object
- Subject
  - Active entity requesting access to an object or data within an object
  - Different subjects have different access levels
  - e.g. users, processes etc.

Access Control

- Concerned with authorisation: what a subject is allowed to do
- Mediates a subject access to object
- Enforces a security policy, limiting which actions are allowed

Complete mediation

- That is our aim
- Trusted computing base: all hardware and software that is responsible for enforcing the policy
  - For example, OS kernel, all trusted processes, and the PC hardware. A fault in one can compromise security (we have seen many examples).
- Complete mediation is important (if it can be subverted, then use is limited)
- Reference monitor concept (all access through it)



Protection state

- Security context: security identity information used to inform authorisation decision.
  - For example, UID and GID associated with a process
- The protection state is made up of the security sensitive actions every entity is able to do
  - For example, when a user change identity the protection state has changed
- Transition between protection state need to be tightly controlled (e.g. setgid)

Access Control Matrix

- Simplest way to represent the protection state of a system
- A table representing every subjects and objects, and the permitted type of actions between them

	File 1	**File 2**	**File 3**
User A	Read, Write, Own	Read, Write	Read, Write, Own
User B	Append	Read, Write, Own	Read, Write

What’s the fundamental issue here?

Access Control Matrix

- Could in theory express any possible access control policy
- Only useful in theory…
- In any practical systems there is too many subjects and objects
- … but any protection state can be expressed this way

	File 1	File 2	File 3
User A	Read, Write, Own	Read, Write	Read, Write, Own
User B	Append	Read, Write, Own	Read, Write

Access control policy

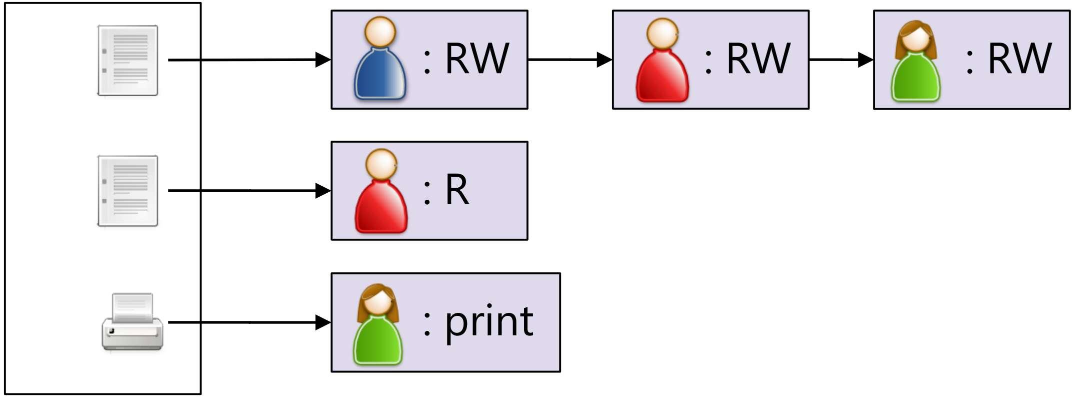
- Express formally or informally what is allowed
- Generally express confidentiality/integrity requirements

	File 1	File 2	File 3
User A	Read, Write, Own	Read, Write	Read, Write, Own
User B	Append	Read, Write, Own	Read, Write

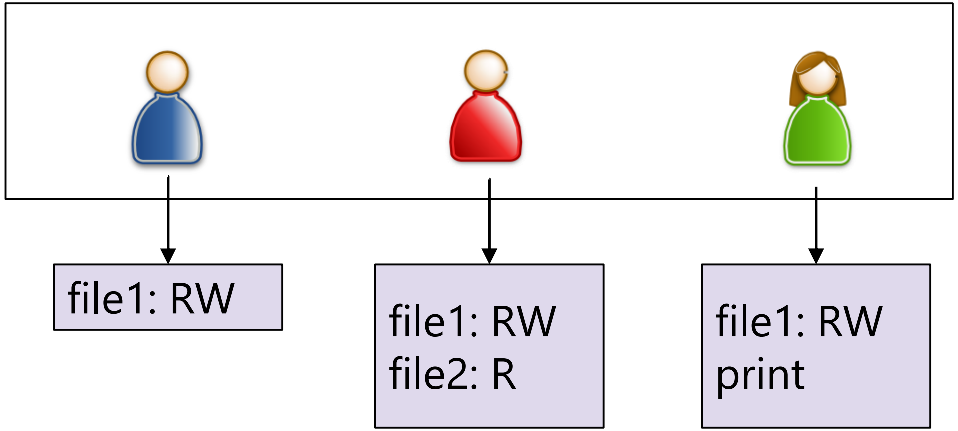
Mechanisms and Models

- A mechanism how the policy is enforced (think of softwared/hardware implementation)
  - e.g. SELinux, AppArmor on Linux distributions
- A model how to represent policies
  - e.g. access control matrix (this is not a good idea)

Access Control List



Capabilities



file1: RW

file1: RW
file2: R

file1: RW
print

Access Control List/Capabilities

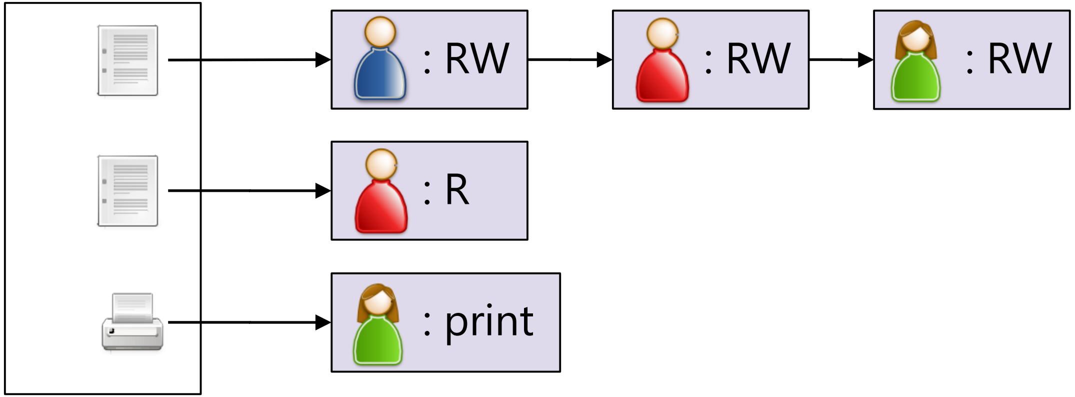
- Access Control List
  - Object centric
- Capabilities
  - Subject centric

Discretionary / Mandatory

- Discretionary: owner of a resource defines/delegates right
  - Requires the notion of owner
  - Most systems
  - Resource owner define associated policy

- Mandatory: policies centrally set by an administrator
  - Notion of owner optional
  - Well suited when an organization own the data (e.g. military government)

Access Control List

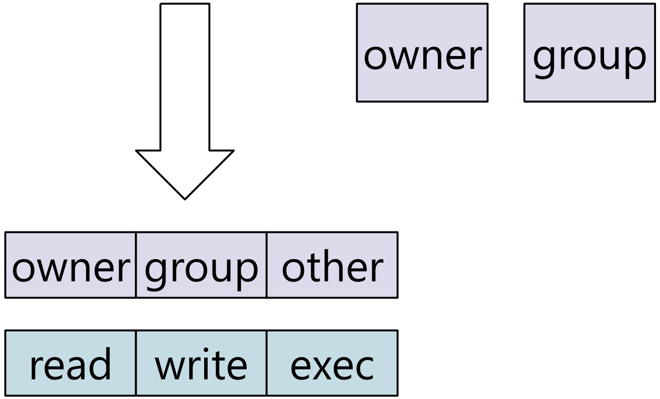


Access Control List

- On Windows work more or less like this (virtually unlimited list)
- On UNIX: 3 entries Owner, Group, World
- By default owner=creator
- Inheritance rules (objects inherit rules of containers)
  - e.g. files inherit folders permission
- Owner can revoke access
- DAC

UNIX

```
~/syssec/2017$ ls -l
-rw-r--r-- csxdb cosc ... accesscontrol.pdf
-rw-r--r-- csxdb cosc ... systems.pdf
-rw------- csxdb cosc ... exam.pdf
-rwxr-x--- csxdb cosc ... slides.sh
```



owner group

owner | group | other

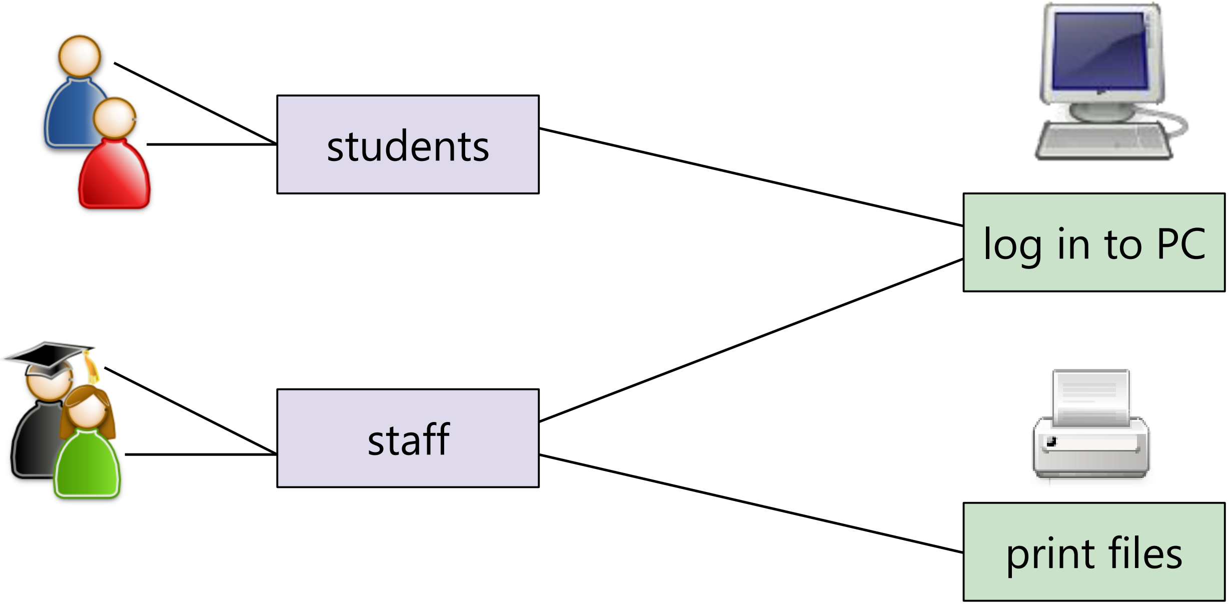
read | write | exec

University of BRISTOL

Role-based Access Control

bristol.ac.uk

Groups / Roles



students

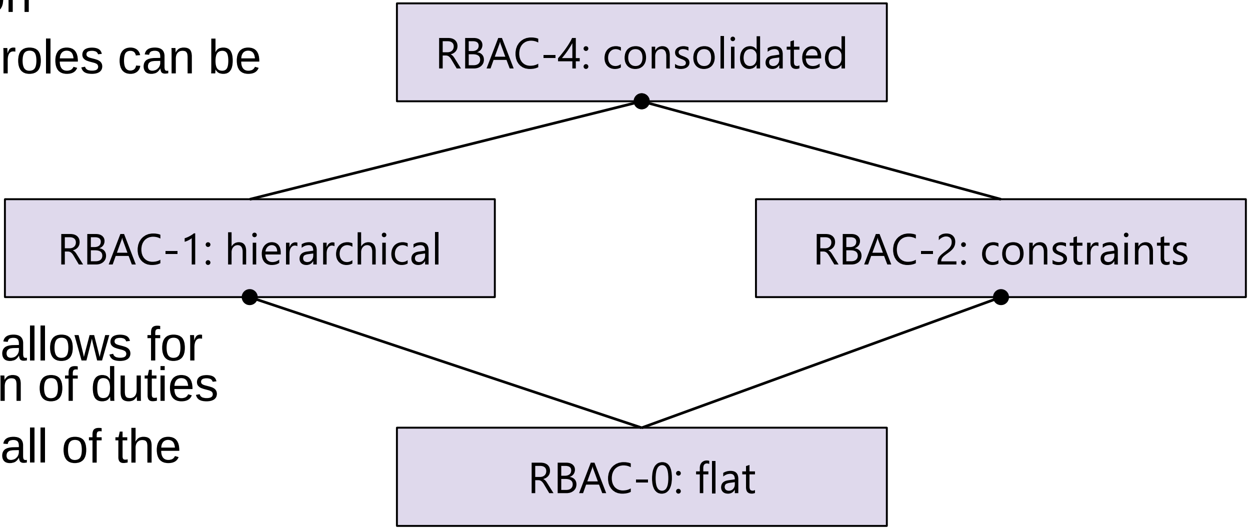
staff

log in to PC

print files

RBAC (NIST)

- RBAC 0: users have a role, role have permission
- RBAC 1: roles can be inherited



- RBAC 2: allows for separation of duties
- RBAC 4: all of the above

Information Flow Control

IFC

- read: receiving information from another entity
- write: sending information from another entity

- Concept emerges from US military

IFC- Bell Lapadula

Top-secret (2)
Secret (1)
Unclassified (0)

IFC- Bell Lapadula

Top-secret (2)

Secret (1)

Unclassified (0)

- read: $f_s(s) >= f_o(o)$
  - no read up

IFC- Bell Lapadula

Top-secret (2)

Secret (1)

Unclassified (0)

- read: $f_s(s) >= f_o(o)$
  - no read up
- write: $f_s(s) <= f_o(o)$
  - no write down

IFC- Bell Lapadula

Top-secret (2)
Secret (1)
Unclassified (0)

- read: $f_s(s) >= f_o(o)$
  - no read up
- write: $f_s(s) <= f_o(o)$
  - no write down
- read/write: $f_s(s) == f_o(o)$

Problem with this naïve approach?

IFC - Bell Lapadula

Top-secret (2)
Secret (1)
Unclassified (0)

- Need to add categories!
- (c, s)
  - c classification
  - s categories
- $f_s(s)$ dominates $f_o(o)$
  - if $c_s >= c_o$ MAC
    - e.g. top-secret >= secret
  - if $s_s \supseteq s_o$ DAC
    - e.g. {Iraq, Syria} ⊇ {Iraq}

IFC- Bell Lapadula

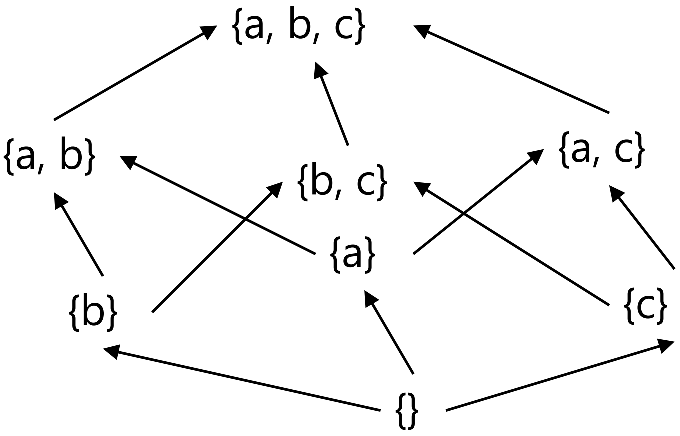
Top-secret (2)
Secret (1)
Unclassified (0)

- read: $f_s(s)$ dominates $f_o(o)$
  - no read up
- write: $f_o(o)$ dominates $f_s(s)$
  - no write down
- read/write: $f_s(s) == f_o(o)$

- SELinux variant
  - Level per category

IFC - Bell Lapadula

- Categories have a partial order.
- Form a lattice



IFC - Biba Model (similar idea for integrity)

Fact (2)
Belief (1)
Rumor (0)

IFC - Biba Model (similar idea for integrity)

Fact (2)
Belief (1)
Rumor (0)

- read: $f_s(s) <= f_o(o)$
  - –no read down

IFC - Biba Model (similar idea for integrity)

Fact (2)
Belief (1)
Rumor (0)

- read: $f_s(s) <= f_o(o)$
  - no read down
- write: $f_s(s) >= f_o(o)$
  - no write up

IFC - Biba Model (similar idea for integrity)

Fact (2)
Belief (1)
Rumor (0)

- read: $f_s(s) <= f_o(o)$
  - no read down
- write: $f_s(s) >= f_o(o)$
  - no write up
- read/write: $f_s(s) == f_o(o)$

IFC - Biba Model (similar idea for integrity)

Fact (2)
Belief (1)
Rumor (0)

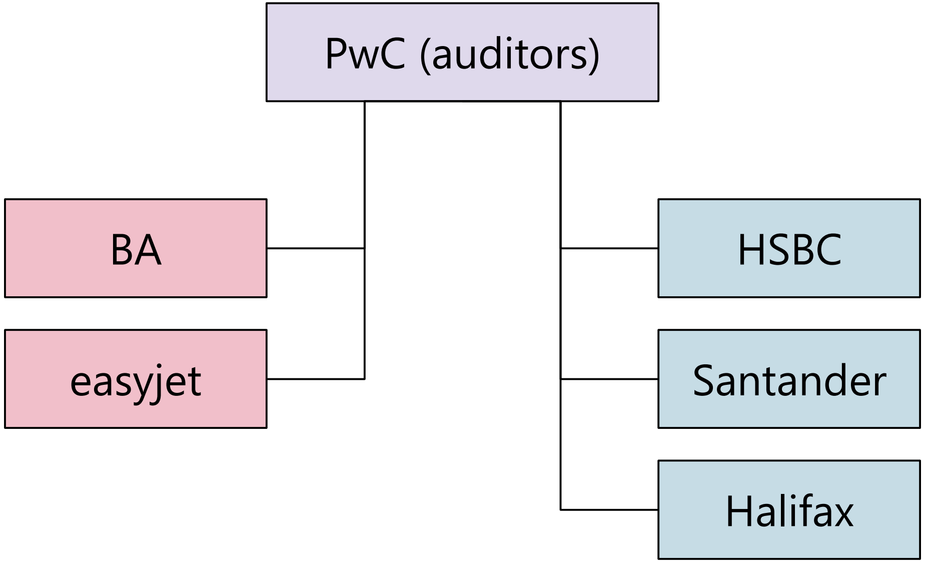
- read: $f_s(s) <= f_o(o)$
  - no read down
- write: $f_s(s) >= f_o(o)$
  - no write up
- read/write: $f_s(s) == f_o(o)$
- Similarly we can add categories

University of BRISTOL

Brewer and Nash Model

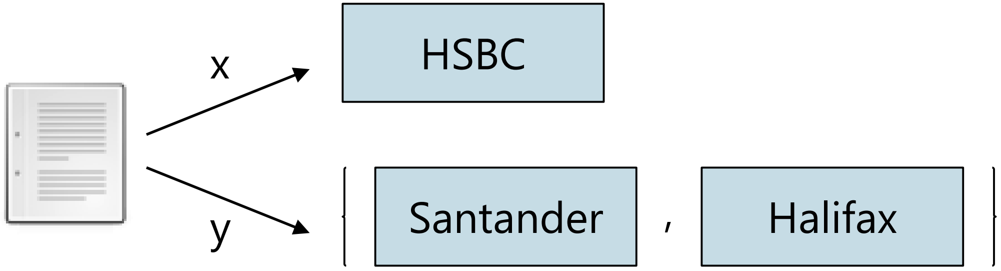
a.k.a Chinese Wall

Chinese Wall Model



Chinese Wall Model

O = objects (clients' files)
C = clients
CoI(o) = Conflict of Interest Group
x(o): owner, y(o): conflict class CoI(o) – x(o)



Chinese Wall Model

- Define n(s, o): has s ever read o?

Chinese Wall Model

- Define n(s, o): has s ever read o?
- s can read o if:
  - For all o’ such that n(s, o’)=true, y(o) = y(o’) or y(o) ∌ x(o’)
  - o is from the same company as previously read objects or o belong to a different conflict of interest class

Chinese Wall Model

- Define n(s, o): has s ever read o?
- s can read o if:
  - For all o’ such that n(s, o’)=true, $y(o) = y(o')$ or $y(o) \not\ni x(o')$
  - o is from the same company as previously read objects or o belong to a different conflict of interest class
- s can write o if:
  - s can read o
  - $\nexists o'$ such that n(s,o’)=true with $y(o) \neq y(o')$ unless $x(o') = \emptyset$
  - You can’t write client data if reading anyone else data, unless other data have been sanitized

Chinese Wall Model

- Define n(s, o): has s ever read o?
- s can read o if:
  - For all o’ such that n(s, o’)=true, y(o) = y(o’) or y(o) ∌ x(o’)
  - o is from the same company as previously read objects or o belong to a different conflict of interest class
- s can write o if:
  - s can read o
  - ∄o’ such that n(s,o’)=true with y(o) ≠ y(o’) unless x(o’) = ∅
  - You can’t write client data if reading anyone else data, unless other data have been sanitized
  - Need sanitization/declassification policy
    - ➢ Simple example owner can declassify objects they own
    - ➢ Authority may declassify audit report etc.

Clark-Wilson Model

Integrity

Clark-Wilson Model

- Bank-like context
- Integrity defined as a set of constraints
  - Data in a valid state when it meet constraints
  - Example
    - YB: Yesterday Balance, TD: today deposit, TW: today withdrawal, TB: today balance
    - TB = YB + TD – TW
- Well formed transactions
  - Move system to one consistent state to another
- Require separation of duty
  - Who certify transactions
  - Who verify constraints

Clark-Wilson Model

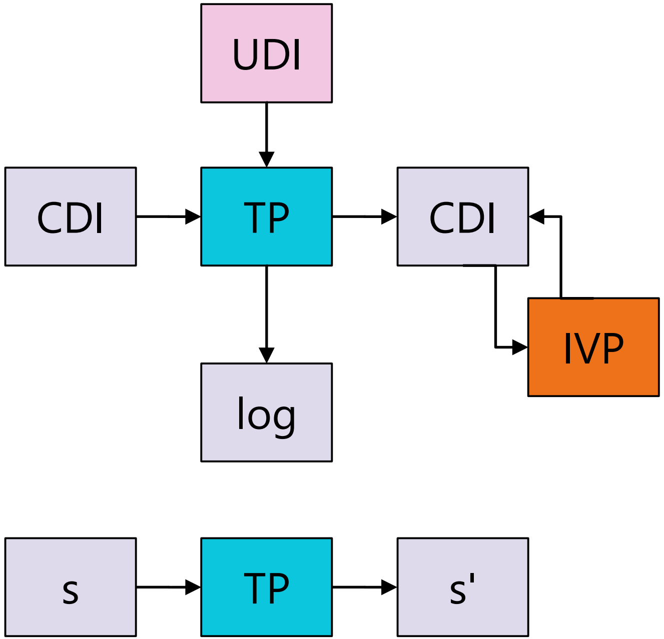
CDI	constrained data item
UDI	unconstrained data item
TP	transformation procedure
IVP	integrity verification procedure

Clark-Wilson Model

CDI	constrained data item
UDI	unconstrained data item
TP	transformation procedure
IVP	integrity verification procedure

- Permission as a triple
  - (user, TP, {CDIs/UDIs})
- e.g. (bob, transfer, hisaccount)
- transfer(CDI:account, UDI:accountref)

Clark-Wilson Model



Certification rules
C1: CDI state validated by IVP
C2: TPs preserve valid state
C3: separation of duty
C4: TPs write to log
C5: UDIs are validated by TPs

Enforcement rules
E1: Only TPs change CDIs
E2: TPs require authorisation
E3: All users are authenticated
E4: only authorised people can change permissions

Summary

- Many models for different objectives
- … and there is many more
  - attribute-based access control
  - provenance-based access control
  - etc..
- Different model for different objectives (in theory you could obtain the same access control matrix from two different models)
- They have led to programming pattern
- In practice hard to implement
  - Real world is full of exception and edge cases
  - May get in the way of business

Thank you

Office MVB 3.26

bristol.ac.uk