Birkbeck
UNIVERSITY OF LONDON

Information Security
Information & Network Security
Lecture 2

David Weston

Birkbeck, University of London
Autumn Term

Birkbeck
UNIVERSITY OF LONDON

Security Policies

Introduction

- So you've succeeded as SO in convincing people that
  - taking care of information security
  - and assessing the risks

  is a good idea
- What's next?
  - Implementing everything and putting things in practice

Introduction (2)

- Implementing information security is not just about scrambling like mad to
  - install the newest patches
  - get firewalls up and running
  - install virus scanners
- It's also about putting sensible practices and procedures in place
  - We're still talking about administrative/management side, we'll look at technical issues later

Security Policies

- This is where the concept of a *security policy* comes into play
- It is a (set of) document(s) in which an organization's
  - philosophy
  - strategy
  - practices

  with regard to confidentiality, integrity, and availability of information and information systems are laid out

Security Policies (2)

- What can policies do that technology alone can't?
  - Tie in upper management: technological controls are the responsibility of an IT administrator/manager, policies are those of upper management
  - If an organization has legal obligations to fulfill, it can provide a paper trail
  - Define a strategy with regard to security: what to protect and to which degree (rather than how)
  - Serve as a guide to information security for employees

Defining a Security Policy

- First step: find out about your risks (risk assessment)
- Policies can be seen as a risk-control mechanism (a procedural one rather than a technical one)
- Goal of a control mechanism is to get risk down to an acceptable level
- Policy should cover risks identified during risk assessment
- Before going into concrete examples, we'll define some important concepts of security policies

Formal Models

- An important aspect of security policies are classification and access control models:
  - Distinguishes between various groups of people, systems, and information in terms of security
  - Determines who is authorized to access which information on which system
- Access Control usually comes in two different flavors:
  - Discretionary Access Control (DAC)
  - Mandatory Access Control (MAC)

Discretionary vs. Mandatory

- Discretionary Access Control (DAC)
  - Each data object is owned by a user and user can decide freely which other users are allowed to access data object
  - Many operating systems follow this line
- Mandatory Access Control (MAC)
  - Access is controlled by a system-wide policy with no say by the users
  - Military systems often use this kind of control
- We'll now look at some concrete models

Access Control Matrix

- One of the earliest models to emerge was the Access Control Matrix
- An *Access Control Matrix* is used to describe the protection state of information or a system
- More formally, there are
  - objects o that need to be protected
    - objects can be files, devices, processes
  - subjects s who want to access the objects
    - subjects can be users, other processes
  - rights $r(s, o)$ that specify which type of access subject s is permitted on object o
    - access rights can include read, write, execute, own

Access Control Matrix (2)

- This is organized in a matrix: subjects are in the rows, objects in the columns, and the rights in the cells

	o_1	o_2	o_3	...
s_1	$r^1_{11}, r^2_{11}, \ldots$	$r^1_{12}, r^2_{12}, \ldots$	$r^1_{13}, r^2_{13}, \ldots$	...
s_2	$r^1_{21}, r^2_{21}, \ldots$	$r^1_{22}, r^2_{22}, \ldots$	$r^1_{23}, r^2_{23}, \ldots$	...
s_3	$r^1_{31}, r^2_{31}, \ldots$	$r^1_{32}, r^2_{32}, \ldots$	$r^1_{33}, r^2_{33}, \ldots$	...
...	...	...	...	...

Example

- Here's a bookkeeping example with different users and objects:

	Operating System	Accounting Application	Accounting Data	Audit Trail
Alice (sysadmin)	rwx	rwx	r	r
Bob (manager)	rx	x	-	-
Charlie (auditor)	rx	r	r	r
Acc. Application	rx	r	rw	w

(r: read access, w: write access, x: execution permission, -: no rights)

Problems with Access Control Matrices

- If applied in a straightforward fashion, this matrix doesn't scale very well
  - Example: bank with 50,000 staff and 300 applications $\Rightarrow$ 15,000,000 million entries
- On top of that, most entries are empty: storing a sparsely populated matrix explicitly wastes a lot of space
- Solution: only store entries that actually exist, can be done in two different ways:
  - by row, store privileges with user (capability lists)
  - by column, store privileges with objects (access control lists)

Capability Lists

- Storing the example matrix by rows would yield the following capability lists:

Alice	Op. Sys.: rwx, Acc. App.: rwx, Acc. Data: r, Audit: r
Bob	Op. Sys.: rx, Acc. App.: x
...	...

- Makes it difficult to check who has privileges for certain files
  - One would have to look at every list

Access Control Lists

- Storing the example matrix column-wise would result in the following access control lists

Op. Sys.	Alice: rwx, Bob: rx, Charlie: rx, Acc. App.: rx
Acc. App.	Alice: rwx, Bob: x, Charlie: r, Acc. App.: r
Acc. Data.	Alice: r, Charlie: r, Acc. App.: rw
...	...

- Makes it easier to check who can access which file (and, if necessary, revoke the rights)
- However, lists may still be too long and it may be too tedious to manage them
- One solution is to organize users into groups

Access Control Lists (2)

- Grouping users is an approach employed by many operating systems
- For example, in UNIX/Linux systems you have the following partitioning:
  - self: owner of a file
  - group: a group of users sharing common access
  - other: everyone else

```
-  rwx  r-x  r-x  alice users accounts
   self group other
```

`alice` is the name of the owner

`users` is the name of a group

`accounts` is the name of a file

Access Control Lists (3)

- However, ownership of a file is still under control of a single user
- This user can pass on access permissions as they see fit
  - Assigning users to groups is usually done by a system administrator, though
- This might not always be appropriate for (larger) organizations
- There is a model called *role-based access control*, which we will look at after introducing some other models

Summary Access Control Matrix

- An Access Control Matrix
  - specifies permissions on an abstract level
  - limits the damage that certain subjects can cause
  - in the form described above is a snapshot view (dynamic behavior is not described)

Bell-La Padula Model

- In the Bell-La Padula (BLP) model, every document (or information object) has a security classification
- The more sensitive the information, the higher the classification level
- Examples for typical levels are: top secret (ts), secret (s), confidential (c), and unclassified (uc)
- Every user of the system has a clearance (level)
- Classification and clearance levels are not decided by the users, some certified entity has to do this
- To be able to access a document, a user must have at least the level of the document
- The concept of this model originated in the military
- Let's define this more formally

Bell-La Padula Model (2)

- Assume that you have
  - a data object o
  - a subject s
- Let $c(o)$ be the classification of o and $c(s)$ be the clearance of s
- BLP enforces the following two properties:
  - Simple security property: s may have read access to o only if $c(o) \leq c(s)$
  - * (star) property: s, who has read access to o, may have write access to another object p only if $c(o) \leq c(p)$

Bell-La Padula Model (3)

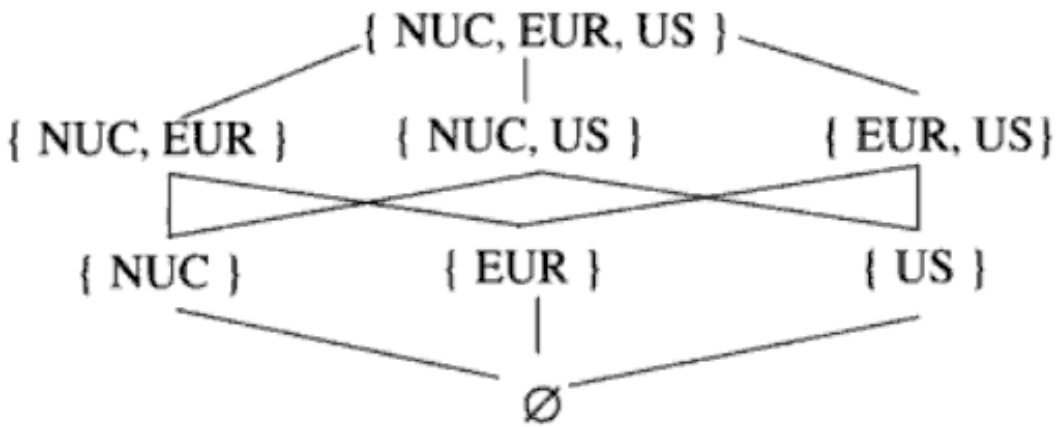
- First rule is quite straightforward: no-one may access information that is classified higher than their clearance
- Second rule seems counterintuitive at first glance
  - Intended to prevent a “write-down” effect
  - “write-down”: more highly classified information is passed to a lower classified object
- BLP can be extended by categories

Categories

- *Categories* (or compartments) describes types of information
- Each data object may be assigned to different categories (multiple categories are possible)
- A subject should only have access to information that they need to do their job
- In addition to a clearance level, a subject is also assigned a set of categories (which they are allowed to access)

Categories (2)

- Let's assume we have the categories NUC, EUR, US
- All possible subsets of these categories form a lattice with regard to the subset relation:



- A person with access to { NUC, US } may look at data objects categorized as { NUC, US }, { NUC }, { US }, or ∅ (uncategorized), subject to clearance level

Summary BLP

- In essence BLP enforces confidentiality
- Can be explained by its military origin
- Does not handle data integrity (or only to a very limited extent)
- We'll have a look at some other models now (which also ensure integrity)

Clark-Wilson Integrity Model

- In a commercial environment, preventing unauthorized modification of data is at least as important as preventing disclosure
- Examples:
  - Preventing fraud (or mistakes) in accounting
  - Inventory control:
    - Still works if information is disclosed
    - May break down if too much data is incorrect

Clark-Wilson Integrity Model (2)

- BLP protects data against unauthorized users
- The Clark-Wilson (CW) integrity model also tries to protect data against authorized users
- This has started long before computers, two mechanisms are especially important
  - Well-formed transaction: user is constrained in the way they can manipulate data, e.g.
    - record a log of changes
    - double entry bookkeeping
  - Separation of duty: executing different subparts of a task by different persons, e.g.
    - authorizing purchase order, recording arrival, recording arrival of invoice, authorizing payment

Clark-Wilson Integrity Model (3)

- How are these concepts translated for data integrity in information systems?
- Identify *Constrained Data Items* (CDIs)
  - These are data items for which integrity has to be upheld
  - Not all data items need to be CDIs, other data items are called *Unconstrained Data Items* (UDIs)
- The integrity policy is defined by two classes of procedures:
  - *Integrity Verification Procedures* (IVPs)
  - *Transformation Procedures* (TPs)

Integrity Verification Procedures

- IVPs check that all CDIs in the system conform to integrity constraints
- On successful completion this confirms that at the time of running an IVP, the integrity constraints were satisfied
- Example for accounting:
  - Audit functions are typical IVPs
  - For example, an auditor confirms that the books are balanced and reconciled

Transformation Procedures

- TPs correspond to the concept of a well-formed transaction
- Applying a TP to a CDI in a valid state will result in a CDI that is still in a valid state
- Example for accounting:
  - A double entry transaction is a TP

IVPs and TPs

- CW works in the following way:
  - We start in a valid state (confirmed by IVPs)
  - We only manipulate CDIs with TPs
  - We will always be in a valid state
- What's missing?
  - A system can enforce that CDIs are only manipulated by TPs
  - A system cannot ensure (on its own) that the IVPs and TPs are well-behaved
- IVPs and TPs need to be certified by someone

Certification

- The security officer (or an agent with similar responsibilities) certifies IVPs and TPs with respect to an integrity policy
- This is usually not done automatically, but semi-automatically (i.e. some human intervention is necessary)
- Accounting example:
  - SO certifies that TPs implement properly segregated double entry bookkeeping

CW Model

- Assuring the integrity in CW consists of two important parts:
  - Certification (which is done by a human)
  - Enforcement (which is done by a system)
- CW consists of a set of rules referring to certification (C-rules) and enforcement (E-rules)

Concrete Rules

- Basic framework:
  - C1: (IVP certification)
    Successful execution of IVPs confirms that all CDIs are in a valid state
  - C2: (TP certification)
    Agent specifies relations of the form (TP_i, (CDI_{i_1}, CDI_{i_2}, ...)) that defines for which CDIs TP_i returns valid states
  - E1: (TP enforcement)
    System maintains a list of relations and only allows manipulation of CDIs via the specified TPs

Concrete Rules (2)

- Separation of duty:
  - C3: (User certification)
    Agent specifies relations of the form (UserID, TP_i, (CDI_{i_1}, CDI_{i_2}, ...)) that defines which TPs may be executed on which CDIs on behalf of which user
  - E2: (User enforcement)
    System maintains a list of relations and enforces them
  - E3: (Authentication)
    Each user must be authenticated before executing a TP for that user
  - E4: (Certification and execution)
    Only an agent permitted to certify entities may change relations, this agent may not have any execution rights with respect to that entity

Concrete Rules (3)

- Logging/Auditing:
  - C4: (Logging)
    All TPs must be certified to write to an append-only log, making it possible to reconstruct each executed operation
- There's no special enforcement rule, because
  - the log can be modeled as a CDI
  - and each TP has to be certified to append to this CDI
- We're almost done...

Concrete Rules (4)

- Relationship between UDIs and CDIs
  - C5: (Transformation)
    Any TP taking a UDI as an input must be certified to
    - only use valid transformations when taking the UDI input to a CDI
    - or reject the UDI
- This is important as new information is fed into a system via UDIs
  - For example, data typed in by a user may be a UDI

Summary CW

- CW enforces integrity:
  - Uses two different integrity levels: CDIs and UDIs (CDI being the higher level)
  - “Upgrades” from UDIs to CDIs via certified methods
- Could also enforce confidentiality (not a main concern in the original model):
  - If users are only allowed to access data through Transformation Procedures as well
  - Although in this case, no data will be changed

Role-Based Access Control

- Role-based Access Control (RBAC) is a mandatory access control model
- Mainly for commercial/civilian applications, so does not have the different levels of secrecy of Bell-La Padula
- Can be seen as a more strict version of the Access Control List (ACL)
  - Main difference to ACL: in RBAC a certified entity grants and revokes privileges, this is not done by the user

Roles

- Similar to ACL RBAC also puts users into different groups, these are called *roles*
- Depending on their current role, users can access (and modify) certain data objects
- For example, in a hospital setting you could have the following roles: doctors, nurses, pharmacists, administrators
  - While doctors may modify a diagnosis file, administrators may not do so
- Let's give a more formal description now

Formal Definition

- In RBAC you have a
  - subject s
  - role r
  - transaction t (transactions access/modify data)
  - object o
  - mode x (read, write, execute, etc.)
- Furthermore, there are
  - $AR(s)$: the active role of s, i.e. the role s is currently using
  - $RA(s)$: the roles s is authorized to perform
  - $TA(r)$: the transactions a role is authorized to perform

Formal Definition (2)

- A subject s may execute a transaction t if $exec(s, t)$ is true
- $exec(s, t)$ is true, if and only if s can execute t at the current time
  - This is determined by $AR(s)$ and $TA(r)$
  - This is the case if $t \in TA(AR(s))$

Rules in RBAC

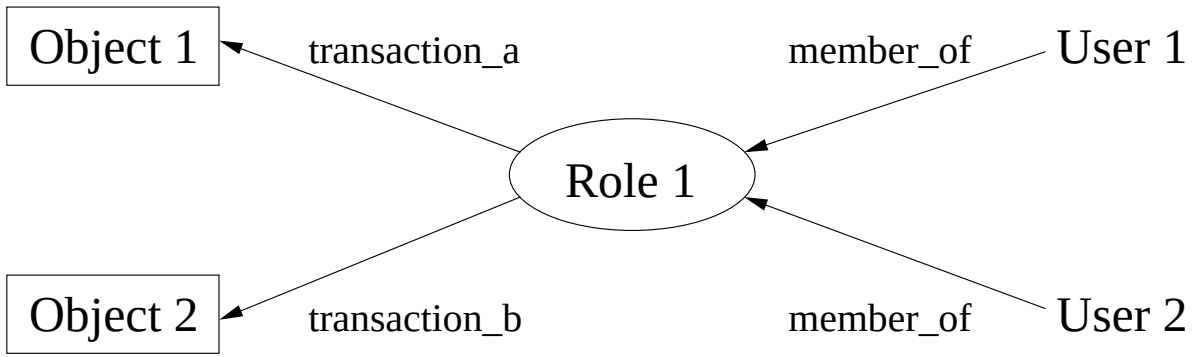
- RBAC enforces the following rules
  1. Role assignment: the assignment of a role is a necessary prerequisite for executing transactions
     For all s, t : $exec(s, t)$ can only be true if $AR(s) \neq \emptyset$
  2. Role authorization: a subject's active role must be authorized
     For all s : $AR(s) \in RA(s)$
  3. Transaction authorization: s can execute t only if t is authorized for the active role of s
     For all s, t : $exec(s, t)$ is true only if $t \in TA(AR(s))$

Rules in RBAC (2)

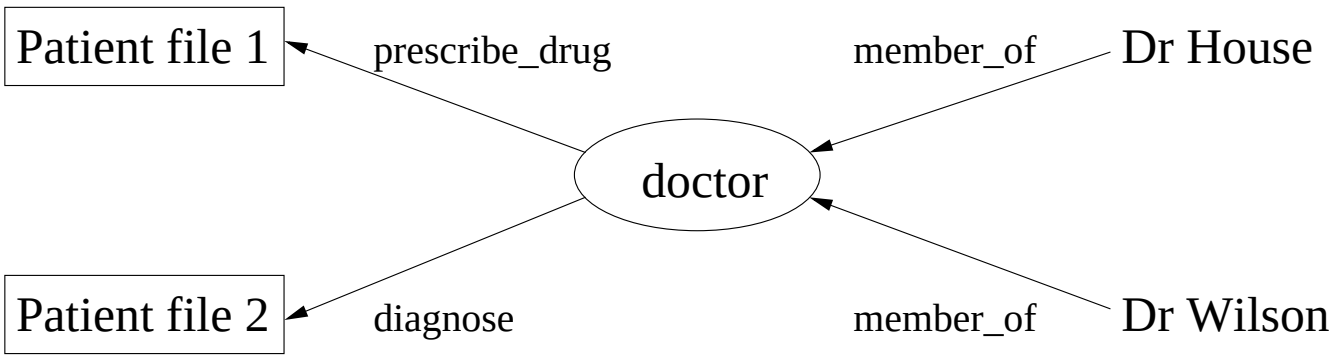
- Access modes can be defined for transactions (to allow a finer granularity)
  - This defines what a user (in a certain role) can do with data (read-only, modify, etc.)
  - $access(r, t, o, x)$ is true, if and only if role r may execute transaction t on object o in mode x
- So sometimes a fourth rule is added to RBAC:
  4. For all s, t, o : $exec(s, t)$ is true, only if $access(AR(s), t, o, x)$ is true

Role Relationships

- The relationship between users, roles, transactions, and objects can also be described graphically



- A more concrete example in a hospital setting:



Summary RBAC

- Often MAC is equated with military applications, while DAC is seen as appropriate for commercial/civilian applications
- However, DAC might not always provide the appropriate security in a commercial/civilian setting
- RBAC fills this gap: it's a MAC model without the specific military requirements

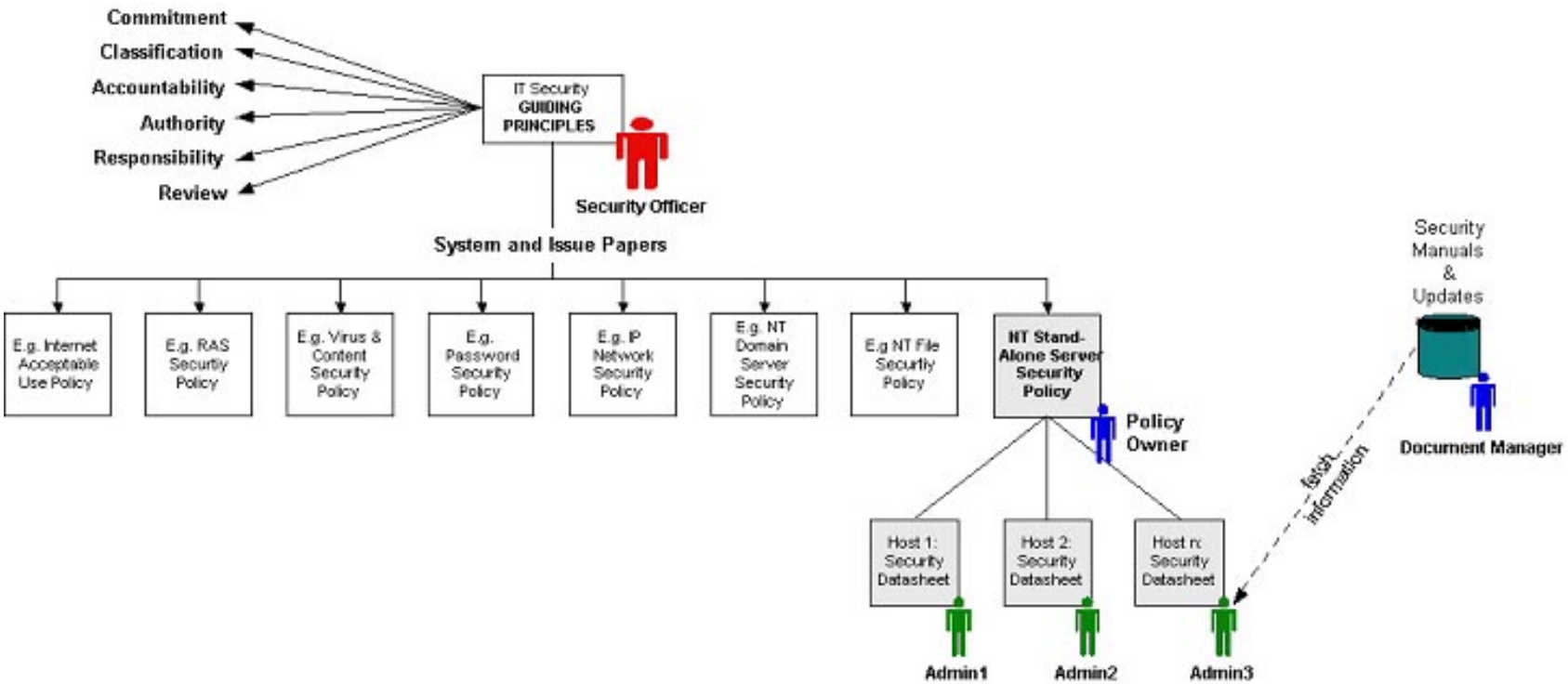
Summary of Formal Models

- Access Control Matrix, BLP, CW, and RBAC are just four (though important) models
- Other well-known models include Biba (integrity) and Chinese Wall (hybrid)
- Due to time constraints they are not discussed here
- We will now turn our attention to the management side of security policies

Organization-wide Policies

- Ideally, a policy for an organization should be
  - light, i.e. not covering hundreds or even thousands of pages
  - simple and practical
  - easy to manage and maintain
  - accessible, i.e. people can find what they're looking for
- Policy should be organized in a hierarchical fashion:
  - Security framework paper
  - Position papers (on specific issues)

Organization-wide Policies (2)



Framework Paper

- The security framework document should
  - state an organization's commitment to information security
  - define a classification system (e.g. based on one of the discussed formal models)
  - make clear that users and administrators will be held accountable for their behavior
  - give the SO appropriate authority to enforce security policy
  - state clearly the responsibilities that individuals have
  - define how security will be reviewed

Position Papers

- Position papers
  - address specific aspects of the security policy, e.g.
    - how to configure servers connected to the internet
    - the process to be followed in the event of a security breach
  - should be focused and kept short
  - can have different authors (should be written by someone with expertise in the area)

Position Papers (2)

- Exact topics depend on the organization, here are some important ones:
  - Physical security
  - Network security
  - Access control
  - Authentication
  - Encryption
  - Key management
  - Compliance
  - Auditing and Review
  - Security Awareness
  - Incident response and contingency plan

Content of Position Papers

- Scope: should describe which issue, organizational unit, or system is covered
- Ownership: name and contact details of the document owner
- Validity: policies can/should have limited lifespans (after which they're reviewed)
- Responsibilities: who is responsible for which elements
- Compliance: the consequences of non-compliance with the policy
- Supporting documentation: references to documents higher and lower in the policy structure
- Position statement: statements about the actual issue or system (the hard part)
- Review: whether, when, and how issue or system will be reviewed

Summary

- Policies are an important part of information security, as they also cover the non-technical side
- There's no "one size fits all", policy must be adapted to the risk profile of an organization
  - There are (customizable) policy solutions out there, so not everything has to be done from scratch
- Security policies should be easy to access, use and understand
- For a more detailed example, see `http://www.securityfocus.com/infocus/1193`