Digital Identities

A personal vision on digital identities in the banking sector

Borja Roux

July 2018

Contents

1 Digital Identity in banking: a personal vision **2**

2 Executive summary **3**

3 Some concepts about identity **4**

4 What is an Identity and an identifier **7**

4.1 What we can do with an ID 8

4.2 What usually happens regarding identifiers 9

4.2.1 Different identifiers are defined on a per system basis . . . 9

4.2.2 The same entity is identified by two different identifiers . . 10

4.2.3 We cannot match accounts in a system with the person who uses it 11

4.2.4 Orphan accounts 11

4.2.5 Accounts for systems or things which have no person in charge 12

5 Identity federation **13**

5.1 Identity Un-federation 14

5.2 Which is a good identifier for a company 15

5.3 What happens when an entity legally belongs to two different entity types 15

6 Identity-classification 16

6.1 The birth of an identity . . . 16

6.1.1 Enrolled identity . . . 16

6.1.2 Integrated or Aggregated . . . 17

6.1.3 Fragmented . . . 17

6.1.4 Profiled . . . 17

6.1.5 Projected . . . 17

6.1.6 Inferred . . . 18

6.2 Identity dimensions . . . 18

6.2.1 Completitude . . . 18

6.2.2 Reliability . . . 18

6.2.3 Fragmentation . . . 19

6.3 Where are the accounts? . . . 19

7 Principles of identity modeling 20

7.1 Some misconceptions about identity . . . 21

8 About centralised control and how it compares to distributed management and auditing 24

8.1 Who is the owner of the information? . . . 24

8.2 Who is in contact with reality? . . . 25

8.2.1 Self service within an organisation and between organisations 26

8.3 Relevance over agility . . . 27

8.3.1 The cost of a mistake . . . 28

8.4 The difference between centralised management and centralised control . . . 29

8.5 Conclusions about centralised management . . . 29

9 Authorisation Model **30**

9.1 Classic authorisation models 30

9.1.1 A deeper look into RBAC 31

9.1.2 A deeper look into ABAC 36

9.1.3 Is there room for MAC/DAC/ACL? 38

9.2 The reason for a complex authorisation model 38

9.3 Propagation of the changes in the authorisation model and the attributes 39

10 The use of an identity **40**

10.1 Identity management processes 40

10.1.1 Obtaining identity information 41

10.1.2 Identity information storing and making available 41

10.1.3 Identity lifecycle management 42

10.1.4 Identity information improvement 43

10.1.5 Identity information retrieval for actual company uses . . . 43

10.2 IDM for employees, contractors, consumers and customers 44

10.2.1 The use of an Identity Manager (IDM) 44

10.2.2 CIAM identity manager 45

10.2.3 Convergence of identity management for internal and external users 45

11 Uses of an IDM **47**

11.1 What are these tasks 47

11.1.1 Identity Governance 47

11.1.2 Identity administration 48

11.1.3 Identity provisioning 49

11.1.4 Reconciliation 49

11.2 Workflows 50

11.2.1 The basis for identity administration 51
11.2.2 The accountability for non-personal identities 51
11.2.3 User privilege cloning 52
11.3 Complexity of the authorisation model 53
11.4 Where entitlements belong 53
11.4.1 There is no IDM; only the system repository 54
11.4.2 IDM as sole repository for authorisation 54
11.4.3 IDM plus another repository 55
11.5 Identity management as an agility enabler 56

12 Authentication, authorisation and access control 58

13 Authentication 59
13.1 Classification 59
13.2 Extending the categories beyond people 60
13.2.1 Something the service knows 60
13.2.2 Something the service is 60
13.2.3 Something the service has 61
13.2.4 How the service behaves 61
13.3 Authentication schema 62
13.3.1 Selecting authentication mechanisms 62
13.3.2 Defining trustworthiness of each mechanism 62
13.3.3 Defining operation sets 63
13.3.4 Classifying operation sets 63
13.3.5 Continuous authentication 63
13.3.6 Environmental factors 64
13.3.7 Business behaviour 64
13.3.8 Authenticating services in a microservice architecture . . . 64

13.4 SSO (Single Sign On) as a flexible concept 65
13.5 Biometric information storage 66
13.6 Authenticating a part of an identity 67
13.7 Talk or share in delegated authentication 68

14 Authorisation **69**
14.1 Where authorisation takes place 69
14.1.1 Within the service 69
14.1.2 Before the service 70
14.1.3 Next to the service 70
14.1.4 Adding models external models 71
14.2 The result of the authorisation 71
14.2.1 Operation modifications 71
14.3 Getting the information for the aurhorisation 72
14.3.1 Two different approaches: OIDC (OpenID Connect) Vs XACML (eXtensible Access Control Markup Language) . . 73
14.4 Improving the time to evaluate authorisation 73
14.4.1 Cahing the authorisation request 73
14.4.2 Cahing identity attributes 74
14.4.3 Cache management 75
14.5 Carrying identity information 75
14.6 The thin line between business and security 76

15 Company relationship with identities **77**
15.1 Identity driven organisation 77
15.1.1 Servant leadership of identity management 78
15.2 Relationship with the identity community 78
15.2.1 Working groups and standardisation committees 79

15.2.2 Professional associations 79
15.2.3 What does "belonging to the community mean" 79
15.3 Who should guide the relationship of the company with the identities? 80
15.3.1 The Identity Team 81
15.4 Compliance is not enough 81
15.5 Standard based Vs Ad hoc is the new Build or Buy 82

16 Business models around the identity 84

17 … and so much more … 86

Chapter 1

Digital Identity in banking: a personal vision

This document contains my own personal vision about digital identities, its management, its value and its use in a banking environment.

I have chosen banking sector because I think there is a “perfect storm” coming in that sector regarding identities: the banking model is changing, the relationship with customers and employees are changing, the normative framework is changing and the threats are changing too. All these fronts must be tackled sooner rather than later, and I think focus on identities may provide a fresh view to set company strategies.

It shows a situation that may be far away from current status for most companies at the moment, and that is one of the purposes of this work. Why would anyone write (or read) a vision that is close to current reality?

Chapter 2

Executive summary

Identities are inherently complex because are abstractions of extremely complex entities (people mostly). There is no right solution to most problems; there are many tradeoffs to understand and decisions to make.

Understanding each of the aspects of the identities enables the definition of a corporative long term vision of the identities, and that vision shall guide the efforts of the company.

The relationship of the company with the community of identity professionals and the internal identity team is crucial for the success of identity initiatives.

Chapter 3

Some concepts about identity

- **ABAC (Attribute Based Access Control)**: Authorisation model based on the evaluation of custom policies based on arbitrary attributes.
- **Access Control**: Act of restricting or allowing access to an operation. In most cases it is the implementation of the decision taking during the authorisation process.
- **Account**: a record of an identity in a system. Enables the access to the actions performed by the system.
- **Authentication**: Process of verifying an identity, normally used in order to grant access to a system or to perform an operation.
- **Authorisation**: Process of deciding whether an operation should be allowed or not, and in some cases what action should be taken.
- **Business rules**: Automatic rules for identity management and provisioning based on identity information.
- **CIAM (Customer Identity and Access Management)**: The discipline that works in detail with the with identities of customers and consumers.
- **Connector**: Part of an IDM that understands a managed system and is in charge of the semantical mapping and the communication between the IDM and the managed system.

- **Dynamic groups/roles**: Groups defined by a filter. Instead of associating users to groups, the association is calculated online based on user's attributes.

- **Governance**: Identity processes related to the analysis of the identity information and its improvement. It is specially relevant for employees and contractors and the authorisation information management.

- **IAM (Identity and Access Management)**: The discipline that works with identities, specially with the security-related aspects. Classically it has focused mostly on user accounts and their privileges.

- **IDM (Identity Manager)**: Piece of software whose purpose is the storage of top-level identities for employees and contractors, as well as the management of the authorisation information from a centralised point. It communicates with the rest of the systems via a hierarchical relationship and the processes of provisioning (to push information) and reconciliation (to get information).

- **Identification**: Providing information about an identity (normally oneself). Normally done in conjunction with the authentication.

- **Identity Federation**: Linking information about an identity from different sources (according to wikipedia). This term is used extensively in the context of SSO, but it is not exclusive to it. The value of linking identity information from different sources goes much beyond an easy login.

- **MAC (Mandatory Access Control) / DAC (Discretionary Access Control) / ACL (Access Control Lists)**: Used interchangeably in this document. The three describe authorisation models where the management is done on a case-per-case basis; without grouping subjects.

- **OAuth**: Open standard for access delegation. Most recent version is 2.0, which is the basis for OIDC.

- **OIDC (OpenID Connect)**: Identity layer on top of OAuth 2.0; enhances OAuth by providing identity verification as well as basic profile information of the authenticated identity.

- **Orphan Account**: account that is not known to be tied to any person.

- **PAM (Privileged Access Management)**: Part of identity management that relates to the management and use of accounts that accumulate much privileges and (in some cases) are not strictly associated to a person performing the operations.

- **Privileged Account**: account with special (administrative) privileges. May be associated to specific person or may be anonymous.

- **Provisioning**: Act of pushing identity information from a centralised source (normally IDM) into other identity repositories.

- **RBAC (Role Based Access Control)**: Authorisation model based on the use of the specific attribute roles.

- **Reconciliation**: Act of retrieving identity information from distributed systems into a centralised point (normally an IDM).

- **SSO (Single Sign-On)**: I'll not follow the standard definition and use this: the ability of different systems with loose coupling to authenticate user actions while requiring just one user interaction to log in.

- **XACML (eXtensible Access Control Modelling Language)**: Standard based on XML that provides a detailed definition of the ABAC model.

Chapter 4

What is an Identity and an identifier

Let's discuss what is an identity for the sake of this document.

As a first step, let's think about identities as some means to identify people. Even when it is possible to define an identity system which may be unrelated to people (e.g. mineral classification), for the purpose of this document, highest level [Digital] Identities refer to people.

Of course, there are important things that are not people, and are still relevant for us, such as the devices used by out users, the buildings where they work, the cities where they live, the privileges these people have, etc. We will treat people as first class entities and the rest of the list as second class ones.

There are some other identities that we must take into account and are often forgotten. Most important ones are the identities that relate to IT systems, and the identities that relate to IoT devices. In these two cases, the management is not related to the management of the identities of people.

Knowing what an identity is, and assuming it refers to the identification of an entity, let's discuss what does a good identifier look like and which are its uses.

Forgive me if I go from truism to truism, but I want the reasoning to go smooth.

- An identifier identifies an entity: My passport ID identifies me, and my passport ID does not identify any other person. It should not even if we share the same name, last name, gender, date of birth and the rest of the elements of information in my passport.

- An entity should have only one identifier of a given type: I may have a national ID card with a different ID than the one in my passport, but I should not have more than one passport. This means that when (not if, but when) this happens (and this is not exclusive to fiction movies), we have a problem to address.
- An entity ID does not change unless the true nature of the entity changes: I can change my name, address, gender and physical aspect and I should remain with the same ID. This means that it is not (just) the information registered about me that defines who I am.

This implies some qualities that a good identifier should not have:

- An identifier should not be based (just) on the information registered: I may change my email address and name and remain being the same person.
- An identifier should not be a composition of the information provided: Hashing a record to create an indexable ID becomes a problem when the record changes.
- If identifiers are provided by different (non-synchronized) parties, the ID space should not be shared: I know it is common sense, but it would not have been the first time when two different offices create documents for different people with the same ID, or two traffic offices register the same plate.

4.1 What we can do with an ID

It can be simplified to just two things:

- Read, Update and Delete operations on an entity: Ok, this is the obvious one.
- Cross information from different sources: This is more interesting. If we have the same ID in two different databases, we can cross the information in them.

What cannot (and should not) be simplified is the set of implications for these two things. It enables sharing information across products, across department and across countries.

This information sharing is not restricted to "after-the-fact" operations. It also enables online authorisation, online information showing for our customers, online and unified management of information, etc.

In case this sounds too good to be true, I'll enforce the word "enables"; it does not mean it happens overnight and automatically; it means it cannot happen smoothly without it.

4.2 What usually happens regarding identifiers

Once we have discussed what good identifiers are and what should not be, let's talk about what happens regarding identifiers in the real world.

4.2.1 Different identifiers are defined on a per system basis

Since it is uncommon to have strict identifier policies when the organisation is created, each system tries to solve the identification problem in the easiest (fastest, cheapest) way.

Some of the typical solutions are:

- If the system uses a directory to store identities, the DN (Distinguished Name) is used as the identifier. This ID is useful just within the repository; it ties the system to the technology chosen for the storage of the information.
- If the application enables consumer self-registration, the email is used. For this choosing to be perfect, we have to assume a person does not register with more than one email (this is something possibly no one has ever done to get a second free month from Netflix ;)), suppose the user will be loyal to his email domain forever.
- If the system deals with people from a single nationality, use the national id card. This may work (apart from some mistakes by public administration) if the company works in just one country. If the company works in different countries, identifiers are different; if a user belongs to two countries problems may arise.
- If the system deals with people who have a passport, that ID is used. This is a good approach, given you can convince your users to have a passport and give you the ID.

- If the system creators feel lonely, an internal identifier is used; this identifier may be a composition of the entity attributes (e.g. first name plus last name), an auto-incremental value of a database, a declaredly local identifier (e.g. my-system-0001), a UUID or some other locally defined ID. If the identifier is not good even locally (e.g. it is prone to collisions) it should be changed; if it is good, but it is local, it is a good candidate for mapping to a global identifier.

4.2.2 The same entity is identified by two different identifiers

This may happen on purpose in the beginning of time (e.g. a user that registers two different identities to get a "sign in" benefit twice), we may know about this at some point in time (e.g. we find that John Doe owns two accounts: one called `jdoe`, that is not used anymore and `john.doe` with the newer policy for ID generation). Maybe this happened because of technical reasons (e.g. a user has two usernames because the system only allows one privilege profile per account) or because of personal preferences (e.g. John Doe requested a new account because `firebolt` seemed to be a cooler ID than `jdoe`).

In either case, the situation is the same: we have a duplication to deal with.

We can do a number of things:

- Delete the invalid value: this would be the easiest option, but it is not always conceivable or convenient (e.g. both accounts are being heavily used).
- Merge the information in just one account: this can be a good option if the system allows aliases for the identifier (e.g. email addresses).
- Register the duplication within the system: create a registry for this kind of scenarios. The creation, maintenance and adaptation of the tools to use it makes it impractical for many cases.
- Register the duplication in the association with a higher-level identity: This approach is available in many IDMs (IDentity Management system), and it has the least impact in the system where the duplication happens.

No matter what action we decide to take, the end must be one of these: either the duplication is eliminated, or it is accounted for and taken into account.

4.2.3 We cannot match accounts in a system with the person who uses it

I have known people who wouldn't believe this could ever happen in a company. Unfortunately the sad truth is there are companies, systems and accounts with this situation: the only way to find out whether the account with identifier `jdoe` belongs to John Doe or Jennifer Doe is by asking them directly.

Of course this only happens in certain companies, in certain systems and for certain accounts. This is nothing more than a minimum percentage of the overall account count.

As in the previous case, the most typical way of dealing with this is by adding a relationship between the person and the account in the corporate IDM. Adding a field for the matching (e.g. a global identifier within the company) may also be a good option if the system allows it.

4.2.4 Orphan accounts

In the previous case, we had an account that belonged to someone and we were not sure who the identifier referred to. In this case, we find an account that has not been used for a long time (if we know when was the last login for this account we are on the good track), and we are uncertain about whether it is used any more or not.

The most common way to deal with this situation is by disabling all accounts which have not been used for a long time (e.g. six months). This does not provide immediate information about who the account belongs to, or whether it is used at all, but solves a possible security hole.

Please beware that this is not a good option in two cases: systems that charge per account (some of them do not charge for disabled accounts), and systems that are seldom used because of their intrinsic nature; I had a really bad time when we disabled accounts in a system that was used to make the annual accounting reconciliation (with emphasis in "annual").

4.2.5 Accounts for systems or things which have no person in charge

Banking is a context where changes to not happen overnight. We all know stories about people in banking who have created quick and dirty programs which were supposed to "last for just a few months until the final solution arrives", and these systems have outlived their creators.

I believe the identity of a system does not have to be subordinated to the one of a person, but there must be some way to govern that identity, and (so far) it takes people to make the governing.

Depending on the company, the maturity in the management of these account varies, but in most cases it is a matter that does not receive enough attention.

Anecdotes go from entire departments that disappear and no one owns these accounts anymore, to production systems that fail overnight because there was some crucial service running with the personal account of an employee who has retired.

This is certainly one of the many aspects of Identity Management that needs to be acknowledged by companies and taken care of by ID professionals.

Chapter 5

Identity federation

Identity federation is the ability to integrate information about an entity from different repositories.

We have concluded in the real world it is quite uncommon to have a common identifier for all systems. This means we need some hack in order for this federation to work.

The options for this hack require a balance between refactoring, strict policy setting and identifier mapping. Let's take a look at them:

- Refactoring: modifying the identifier of a system in order to use a more global identifier. This requires de definition of a global identifier across the company and the modification in the target system (which is not always possible).
- Strict policy setting: We can set a policy to infer an identifier from the information about a user. A typical example of this is the use of the first letter of the first name plus the last name as the ID for the email. This may work until you have collisions and you have people changing their name (e.g. getting married). This policy must be carefully chosen and is almost never a perfect solution in the long term.
- Identifier mapping: Establishing a mapping between a global identifier (or an identifier in a different system) and the value of the identifier in the target system. The cost of this solution falls in between the other two, and usually well prepared for the changes in the future.

Once we have some means to traverse repositories in the same way we could traverse tables using SQL foreign keys, we're prepared to make use of the information.

I think what really gives value to the identities is the relationship between them. When crossing information across repositories we do not only enrich individual identities; we also increase the relationships between them.

This enrichment is such an army knife (it can be used for so many things) that in many cases the initiatives to federate identity information across the company lacks a specific sponsor (with enough weight in the company as to make it possible). Some cases it happens because there is no single and clear use case and sometimes because it requires aligning too many actors.

5.1 Identity Un-federation

There are times when we do not want the information of a repository to be federated with other repositories.

This is more common than we may think:

- We provide identifiers to third parties for them to refer to our identities (normally employees or customers), but we do not want them to know who is the person behind the identifier.
- We fear finding a credit card database dump in the deep web.
- We want to execute some work on a database that cannot relate the information with real subjects (e.g. database dumps for software developers)

In these cases (and supposing the company wants to maintain the ability to make the matching with actual identities) most common options are:

- Create a database to store the mapping of the identifiers.
- Use some FPE Format Preserving Encryption algorithm and store only the key.

In the first case, there is a new database to manage (and backup, and make available, etc) and the mapping may be updated at some point with a limited impact; in the second case there is only one element of information for all the mapping, which is easier to manage and more troublesome when lost (or shared).

5.2 Which is a good identifier for a company

Taking into account all this, I personally believe a company should have its own global identifier, unrelated to any element of information of the entity, using the same address space for all kinds of identities (employees, customers, devices, services, etc). UUID seems to be a good starting point.

May be this is sounds overkill; I can hear voices saying it does not pay off having a 128 bits identifier for a group when all the information about the group is the name, when the name is as short as `admin`, when the name of the group is unique for the whole system, when spelling a UUID is annoying, given you have to externally map the identifier for systems that do not allow the use of custom identifiers, etc. Even in this case, I would stand for a global identifier.

5.3 What happens when an entity legally belongs to two different entity types

We are not going to see an IoT device that is also an employee of the company (at least not in the short term), but we can find internal employees who have been external contractors, and at the same time consumers of our services and paying customers.

In these cases I would suggest to have one single identity, one single identifier, and as many accounts as they are needed.

Chapter 6

Identity-classification

The typical classification of identities in IAM usually refers to the distinctions between employees, contractors and customers; in some cases service accounts receive some attention, and so may do anonymous privileged accounts (e.g. root accounts in unix systems).

This vision tries to go beyond this classification and expands identity considerations to a broader context than accounts. Accounts are the means that IT systems use to associate operations to identities. Identities transcend accounts, have a broader scope and do not have to be associated to any account in any IT system in the company.

6.1 The birth of an identity

Identities are created in may different ways. In most cases they are derived from other identity information.

These are the most common ways of an identity to be created.

6.1.1 Enrolled identity

At some point, the information of an account comes into the scope of the organisation. It may happen because of some typist (or a person) introduces the information by hand, or because it is imported from another system. This is the beginning of the identity journey in the organisation.

6.1.2 Integrated or Aggregated

Identity information merged from the aggregation of different sources of information. e.g. crossing employee records and customer records, crossing information about concrete accounts and privileges in different systems. These identities are mostly used out of IAM discipline; within IAM their use helps in the optimisation of the identity governance.

It is crucial to have some means to relate identities in different sources of information (e.g. well chosen and common identifiers).

6.1.3 Fragmented

Identity information where the scope is well defined by a set of elements of information, but where the system is prepared to work with incomplete information. e.g. Identities of a system where the scope of the records is name, email and telephone, and accepts that some identities have just one of the three fields populated. I have seen little use of these identities in the classical IAM realm; their use for marketing, fraud analysis, etc is doubtless.

6.1.4 Profiled

Profile defined from different sources of information. e.g. hacker profile definition, regulator psychological profile, market segmentation, etc. I think this kind of identity is extremely underused in IAM; it would have huge impact in many areas, such as identity governance.

6.1.5 Projected

Identity created with a subset of the information about another identity. e.g. email account created based on the information in HR system. It is common to project the identity information in a system in order to create an identity in a different system. As a matter of fact, this is the way in which most IDMs strategies work: An account is created based on the information in another system (e.g. centralised IDM, HR database) and additional information is added afterwards.

6.1.6 Inferred

Inferring an identity consists on creating an identity based on the information of different sources that suggest that this identity should be created. This is common in the processes related to role rationalisation, where roles are created, merged, etc in order to have a better model for identity governance.

The true source of information for identity inferring is the relationships of the identities between themselves.

6.2 Identity dimensions

In order to understand the nature of our identities, we have to know the dimensions where they can be measured.

These dimensions (along with the former classification) constitute the basis that will guide us in the use of the identities.

Of course the values of the same identity for different uses may differ. An identity that may be more than enough for the authentication of a user because it has username and password, may be not enough for a marketing campaign because it lacks email address.

6.2.1 Completitude

The classification of a identity in this dimension is made by answering the question “Does the identity have all the elements of information that I need?”.

6.2.2 Reliability

There are many questions to evaluate this dimension: “Does this information come from a source that I trust?”, “Has it been altered?”, “Is the information up to date?”, etc.

6.2.3 Fragmentation

This dimension deals mostly with the value of the information in the identity. The more identities I have merged or accessed (possibly from different sources), the more value I have been able to infuse into this identity.

6.3 Where are the accounts?

If you have read this section up to this point you may be wondering “Where are the accounts, the roles, the groups, the entitlements, etc?”. The answer is simple: “All of them are identities.”.

I’ll give some examples: an Active Directory account that is created because of some automatic process in the IDM is a projected identity. Some groups in Active Directory (which have been typed by some application owner) are enrolled identities. The information provided to a user that comes from different sources (e.g. an aggregate balance of a customers who owns different kinds of assets) is an integrated identity.

The vision in this document does not ignore identities implemented as system accounts; it tries to cover a broader scope. For example, the identity information that is stored in the HR database constitute many identities; even when that information (while it is just stored in that database) cannot be used to “log in” a system.

Chapter 7

Principles of identity modeling

- **Persons are the paramount element of identity systems**: Either directly or indirectly, identity systems relate to people. This makes it unavoidable to manage the information with the utmost respect to the ethical and regulatory concerns. From a practical perspective, even when the entity object of the work are not persons (e.g. an application for Role definition improving), the chances of missing the true point of the work are much higher if we remove people from the equation.

- **Model reality**: Do not force identities to accommodate your model. Expand your model to accommodate them.

- **Reading and reconciling over writing and controlling**: Watch the status of the identities and ask for information about why do they look the way they do. Do not try to impose restrictions until you understand the nature of the identities in a particular context.

- **A static repository schema is never enough**: It does not matter how much time you take into analysing and understanding current needs of a system. When a system goes live and reality knocks to the door of the system designers, options are making reality fit into your model (which is discussed in its own principle), or helping your model fit into the reality.

- **Entity relationships are the true source of value**: Just in the same way as characters in a novel are nothing but a description until they interact with the rest of the entities in the world, the value of entities in a system is nothing compared to the value of their relationships.

- **Identities change over time**: Change is one of the few constants in the universe. This is also true for entities. Creating a model or a system that thinks of entities as something that does not change over time is going to become a problem at some point. Even if the work relates to historical information, it is possible to find elements of information that enhance existing entities, or it may be our understanding of the entities that change. In either case, entity-related systems should acknowledge for entity updating.

- **Specific purpose for information elicitation**: Apart from regulatory-related needs (e.g. GDPR), this principle states the need for separating different uses of the identity information. In the same way as it is unethical to sell to third parties information that has been requested for sending company notifications, it is unethical to use for marketing purposes information that is elicited for fraud prevention (or vice versa).

- **Not all entities are concrete**: In security, it is common to use specific information about an identity as the sole component for identity systems (e.g. a user requires an account in order to access a system). In marketing it is much more common to work with profiles of customers instead of specific customers. This difference is not as much related to the context (security vs marketing) as it is related to the purpose of the work. Just as marketing often work with actual specific customers to understand reality and create better abstractions (profiles), it must be common practice to analyse concrete subjects (e.g. cybersecurity criminals) in order to create attacker profiles and use those profiles for planning actions and countermeasures. This is currently a trend in cybersecurity which should be encouraged.

7.1 Some misconceptions about identity

I'd like to present some of the ideas that at some point seem to have been fixed in the collective mind about identities and their management.

- **Identities is something security team does on their own**: Even when most of the IAM (and CIAM) initiatives are conducted by the security team of the company, the use of identities transcends security and hence it should not be just "something that security people do". This idea comes from the heavy use of identity governance for authentication and authorisation.

- **Identities is something IT does on their own**: This idea about identity management is broader than thinking about it as a security matter, but still does not grasp the full scope of the identities. Information about people (specially customers) guide most of the business decisions we take; hence identities cannot remain an IT concern. It is business who must guide the way identities are modelled and used.

- **Identities are the elements of information used to login**: It is the other way around; the information used to make login belongs to identities, but it is not true that all the identities are somehow related to authentication, authorisation, access control and other security-related operations. For example, the information in marketing databases contain a good deal of identities, and that information is not directly used for logging in.

- **Silver bullet**: As it happens in many (should I say all?) disciplines, at some point some really good idea comes into scene, and that idea seems so good it will solve all the problems. One of the most interesting things I have learnt about this comes from Gartner's hype cycle; any emerging idea gets a high hype, then there is disillusionment, then we learn about what can we do with it, and in the end we use that thing for its purpose and stop trying to use it for anything unsuitable. Many things come to my mind as supposed silver bullets: "Role mining" (part of entitlement analytics), "Business rules" (automatic provisioning based on person information), "ABAC", "Dynamic roles", "OIDC", "Fully centralised management of identities", single corporative directory, etc

- **One best solution**: Identities is a really complex topic. It tries to model an extremely complex world into as few abstractions as possible. If we were not so used to this kind of abstractions, we would think that modelling a person (with all her complexity) into a username (of up to 8 characters) and a password (with its uppercase and lower case letters) is some kind of joke. The "solutions" in identity management are always a tradeoff where you lose much in exchange for (some of) the value you seek, while trying to maintain complexity at a reasonable level.

- **Linear growth of complexity**: Identities take their value from relationships (e.g. user-role, person-person, etc). Relationships in a network (at least most of the ones related to identities) grow exponentially as the number of nodes grow linearly. This means the complexity of managing one thousand identities is not half complex as managing two thousand. It is way simpler. This also means that if identity modelling is not done right, the impact of generating "too many identities" is exponential to the number of identities created.

- **Single source of truth**: {Read with an epic voice (Morgan Freeman if possible)} This is the legend of the identity source of truth (a.k.a. fountain of youth). Some said it was a single source, some others said it was a composition of the HR information with the departmental spreadsheets for the contractors, some added some text files for the service accounts. The source was the beginning for all the identity knowledge. It was said to contain all the information that any IT systems could ever need. With the information in the source, plus a set of simple business rules, all the user accounts in the company could be [re-]created.{Epic voice off}. IMHO and based on my experience, there are many sources of truth for identities; regarding corporate accounts (e.g. employees, contractors), it is possible to have a kickstart with the information from HR plus some excels plus some business rules, but they are absolutely not enough to make a whole identity management system for two reasons: the modelling of identities leaves a huge amount of information out of the picture (specially operational information), and the scope of each source (yes, there are many) is too narrow (e.g. employees, devices, customers, prospects, threats) because the department creating each source of truth is focused on one single type of identities. Trying to create a single source of truth is forging a silver bullet.

Chapter 8

About centralised control and how it compares to distributed management and auditing

It is very common to discuss governance in terms of centralisation vs distribution. Instead of offering a simple answer, I'd like to provide the dimensions that should be analysed for decision making.

In this particular case, I am referring to identities and their information, but the same approach could be used for other scenarios.

Even when there may be a grey scale between a fully centrally controlled system and a fully distributed one, I'll analyse them as complete alternatives.

8.1 Who is the owner of the information?

In this context ownership I means liability. Another way of stating this question is: Who is going to suffer when a mistake is made regarding this information. The mistake will always happen at some point; it may be disclosure, failing keeping the information to update, lack of availability, etc. Suffering may mean different things: paying a fee for violating a regulation, managing sues from upset customers, delaying a commitment, etc.

If the owner of the information is an specific actor, probably the management of that information could be made in a centralised way. If some of the liabilities refer to different actors, we probably should have a shared control over the information.

8.2 Who is in contact with reality?

One of the best books I have read about complexity in organisations (and at this point I hope you're with me into believing identity is a complex matter) is "Organize for Complexity: How to Get Life Back Into Work to Build the High-Performance Organization" by Niels Pflaeging (this). One of its principles is that big organisations end up having a group of people in contact with reality and dealing with it, and a group of people afar from reality and managing the organisation.

Best decisions are made by people who know reality. Simple as this may sound, it is astounding the number of situations and organisations where this principle is violated.

One of the twelve agile principles says "The best architectures, requirements, and designs emerge from self-organizing teams.". This happens because the team who is implementing a software is very much into reality. That reality (of course, combined with proper education and experience) is key in the decision making process.

In identity management, it is quite common to have a centralised team making decisions about the authorisation related to identities throughout the organisation. Obviously that team does not have deep knowledge about the reality in every system these identities are used into. Does this mean this team cannot provide value to the identity management processes? No.

Reality has a tendency to being complex and multidimensional. In most cases, identities are related to many external reality dimensions. Let's list a very few of the "realities" for the management of the authorisation-related information about identities:

- **Incompatible privileges within an application**: we have to make sure no single person has the ability to issue bills and accept them.
- **Incompatible privileges between applications**: no user of an application should be administrator of the underlaying database.

- **Recertification**: external regulatory constraints makes it compulsory to review the privileges an employee has and validate all of them are needed in order for her to perform her job.
- **Enabling privileged access**: some systems do not [easily] accept user management for some privileged operations. These systems still need to be managed and in order to do so, some anonymous credentials must be used.
- **The organisation has to enable the right to be forgotten**: the organisation has to manage the operations required to ensure the right to be forgotten is implemented throughout the company information records.

These four portions of reality may be managed by four different teams within an organisation: the first one can only be managed by someone with very deep knowledge about the application (possibly the team administering the application); the second one cannot be managed unless there is a team with broader sight than one single system (here comes the need for an identity team); the third one must be managed by whomever knows what the employee does to service the organisation (possibly with help from the identity team); the fourth one can be managed on a per-team basis with many gaps regarding audit, or may be managed in collaboration with a team that sets into place a PAM (Privileged Access Management) tool; the fifth requires much knowledge about the organisation, their systems and their management, which can only be achieved with cross and specific teams.

As we can see, much of the management cannot be made without the help of a team that transcends one single IT system. That team is what I'll call the Identity team.

8.2.1 Self service within an organisation and between organisations

One of the contexts where we feel very well suited regarding identity (and privilege) management is in the self-service context.

What mesmerises me about self-service management is the perception of freedom. For me it looks like a puppet show where the threads are so thin (yet strong) that they are not seen by the public. When a system enables you to make the decisions about the authorisation, it makes so under a very strict rules that narrow the ability of the user in an extreme way. It gives the user an extremely small realm and makes her think she is sovereign of the whole world.

There are two important (and broad) requirements for a "self-service" implementation to be successful:

- **The set of administration rules must be very tight**: What is the scope of the user must be very well designed so that she cannot go out of it and still retain all the freedom she needs.
- **The responsibilities for the management are completely segregated**: The liability of the service provider is zero (or next to zero). I cannot think about suing google if I happen to make public a private document about my health. I cannot do it [because of the dreaded EULA and] because it has been me who has made the mistake.

When we are talking about different organisations, it is [relatively] easy to satisfy both requirements; when the service provider and the user belong to the same organisation, there is more room for shortcuts; for not-so-tight administration rules and not-so-complete segregation of responsibilities.

I personally believe self-serviced administration of identity information (authorisation-related) is one of the many solutions that may balance the tradeoffs of identity management in many cases, but it is not a silver bullet. I don't hope it to work in all contexts within an acceptable level of effort and time to be properly implemented.

8.3 Relevance over agility

This tradeoff refers to the balance between the possible consequences of a proper management of the information versus the need to manage it in a speedy way.

If the information is truly relevant (e.g. it enables financial fraud or exposes the company to regulatory sanctions), speeding up the information management process may be not so important, and a centralised management system may be used.

If the information is not extremely relevant for the company (e.g. updating the physical address of a customer), the operations may be done in a distributed way.

8.3.1 The cost of a mistake

One of the tradeoffs to make in identity management regards the cost of an error, and who pays for the error.

In case an error is cheap and if it is not the company who suffers the consequences, we will not pay as much attention as if the error hurts deeply the company.

Paying attention to the possibility of a mistake in the identity information may mean different things:

- **Setting a dedicated team to the task**: we can assume the user is going to make mistakes, so we prefer to dedicate a professional team to the task. Who may make mistakes too, but hopefully they'll happen seldom.
- **Implementing automatic controls**: whenever possible (and feasible), we can implement automatic checks to ensure the information being managed is correct. We will not spend the same amount of resources when checking a target account number for a money transfer (where we may check whether it is blacklisted or not), than when processing an online enrolment of a new bank customer (where we may check whether she is customer of a different bank, confirm the postal address, double check the identification card, etc).
- **Implementing manual controls**: independently of whether we have implemented any number of automatic controls, we decide the information has to be manually checked. The checker may be a professional with specific knowledge about the business (e.g. we must make sure proper deletion of all the records of a person who has exercised her right to be forgotten), or about operations (e.g. we request the boss of the employee who requests access to a document, to verify it is required for her to perform her job).

A little note about the use of the word mistake in this context: it is no different for me that a mistake happens on purpose or not. I understand the risk of a mistake may be calculated (in a very basic way) as the impact of the mistake times the probability of it happening. If some kind of "mistake" can be triggered by internal fraud, of course it may have a higher chance of happening, a higher impact and hence it should have a higher investment for it not to happen.

8.4 The difference between centralised management and centralised control

Centralised management of information enables the decision making for every single situation that may happen. This means that the control is exercised on a case-per-case basis. All actions taken have been validated by the centralised system.

Centralised control of information (without centralised management) delegates the decision making to external (not part of the centralised one) parties. In this case, auditing is set up so that the centralised system is able to know the details about the status of the information. Actions are taken without the knowledge of the centralised system; only corrective actions may be taken, and these happen after-the-fact.

In the first case, the creation of guidelines for the decision making is a good practice that helps with the job, but is not mandatory. In the second case, it is mandatory to set a framework for the work of the external parties. Without that framework, there would only be personal (individual) opinions.

8.5 Conclusions about centralised management

Well, I am afraid centralised management is not a silver bullet, but it is extremely useful in many cases. It is important to understand which are the tradeoffs we make, and evaluate the different options at our disposal.

Even when centralised management is not a good option for a company, it is vital to have an identity team. The analysis of the identity team happens in its own section.

Chapter 9

Authorisation Model

Even when this document tries to be applicable to identities independently of the use of them, this section refers specifically to the work involved in order to use identity information for authorisation, which by definition is a security concern.

9.1 Classic authorisation models

Just in case the reader is not familiar with the classical authorisation models, let's quickly review them:

- **MAC (Mandatory Access Control), DAC (Discretionary Access Control) and ACL (Access Control List)**: These authorisation models are based on associating specific individual identities and entitlements (operation privileges on an object).
- **RBAC (Role Based Access Control)**: Authorisation model that extends the previous ones by associating users (identities) to roles (which are identities too) and roles to entitlements.
- **ABAC (Attribute Based Access Control)**: Authorisation model that extends the previous ones by associating identity attributes to authorisation policies and the authorisation policies to entitlements.

9.1.1 A deeper look into RBAC

In this context we will think about a role as a group of entitlements, and an entitlement as the ability to perform a certain operation on an object. Whether an entitlement requires the creation of a user account in a system or the inclusion of a group in an existing account are details I am not going to deal with in this document. I'll count

Roles have classically been classified in many different ways depending on various criteria: if all the entitlements belong to the same system, the role may be a group; if the role contains entitlements of different systems, it's called a technical group; if the role contains a set of roles, the role is called "business role". I will not make any distinction for these regards. I think it is not necessary for the purpose of this document and it would complicate the reading.

RBAC is the most used authorisation model. It provides a balance between complexity and flexibility that suits most contexts. However, implementations of RBAC are differ from system to system.

In the following sub-sections I'll explain what characteristics must be present in a good implementation of RBAC.

9.1.1.1 Business rules

Business Rules are (as called by some Identity suite vendors) policies that automate the association of users to roles based on identity information. Examples: When a user moves to the sales department, it gets role "CRM-user", that provides her entitlements to access and manage customers in the CRM tool.

A good set of business rules will automate much of the process of providing entitlements to users; they are specially useful when a change to an identity (onboarding, department change, promotion, etc) happens.

A good implementation of the business rules should also have the following characteristics:

- Monitoring of changes in the identities' attributes to know when do they have to be triggered online
- Be applied to more than user-role relationships: modify other attributes, detect and alert inconsistencies in data, etc

- Detect when the rule no longer applies and its effects should be reversed (e.g. provided roles should be removed)

9.1.1.1.1 Dynamic roles and groups

Depending on the role and business rules implementation, it may be useful to count on dynamic roles or dynamic groups (I'll talk just about dynamic roles for simplification).

Dynamic roles are theoretically the same as business rules defined in a reverse way. A business rule states a condition and an association between a user and a role (e.g. users who belong to department sales, must have role r_sales); a dynamic role is a calculated role, and the calculus is done using a condition (e.g. the users who belong to role r_sales are those who belong to department sales).

From a practical point of view, and taking into account the implementation implications of dynamic roles, they make sense within a system, where the definition of the role (or group) can be done by a search filter (LDAP) or a query (SQL).

9.1.1.2 Business processes

Business process (related to identities) are the processes that state who can request permissions (roles in our case), who must approve that request, etc. Business processes tend to be extremely complex and a source of headaches for identity professionals.

I have seen two different approaches to business process (implementation): "try to make it as simple as possible" and use it as an audit trail, and "try to make it as exhaustive as possible" so it contains all the conversations that can possibly happen in the process.

Both approaches are based on the assumption (or the hope) that the business processes will not change (possibly because it is difficult to find budget for one IAM project, and "challenging" to find budget for a second one). I think both approaches miss the point. I believe in Martin Fowler's mantra "if it hurts, do it more often" (here). I think business processes evolve and their implementation must evolve too. Of course I think it is best to begin with a simple implementation and iterate over it, but the iterative evolution must be driven by feedback taken from use, not just interviews for a complex analysis in a waterfall development.

9.1.1.3 Attributes in the user-role relationship

When a user is associated to a role, that relationship may have attributes.

In case it is not obvious why this is mandatory, let's analyse the modelling of this authorisation scenario:

- There is an entitlement that enables a user to approve a mortgage. We'll call it e_mort_a
- There size of the mortgage that the user can approve is not the same for all users
- There is a role associated to that entitlement. We'll call it r_mort_a
- When a user is associated with the entitlement, the size of the mortgage that can be approved by the user is set.

In order to model this situation, we can decide to create many entitlements: r_mort_a_10K (for mortgages of up-to 10 K€), r_mort_a_50K, r_mort_a_100K, r_mort_a_200K, r_mort_a_500K, r_mort_a_1000K and r_mort_a_full (for mortgages of any amount). This will solve the problem of having one attribute in the relationship by creating 6 additional entitlements and 6 additional roles.

If at some point that same application decides that we will have different user profiles, and some of them will go door-to-door selling, but we do not want our sales people to have the same limits when going door-to-door and when approving mortgages in the office. We also decide that we do not want them to have the same amounts during office hours and out of office hours. In order to cope with this situation we will create 7 entitlements for out-of-the-office during- office-hours and 7 entitlements for in-the-office out-of-office-hours. We will have to create roles for all the possible combinations of these entitlements (2401), and this is the moment when we will remember that complexity does not grow linearly, but exponentially in identity management.

When this kind of complexities arise, the number of entitlements and roles grow exponentially unless we have attributes in the user-role (and user-entitlement) relationship.

9.1.1.4 Stating and storing reasons for user-role relationships

We must know why a user is associated to a role.

Whenever a user is associated to a role, there is a reason for it to happen. It may happen because some senior director decided it was required, because the employee himself decided he wanted it, because the association was triggered by a business rule when the user moved to this department, etc. In any case, the reason is important.

At some point, someone (possibly an external auditor) will ask "Why does this user have this role (or entitlement)?". At that moment, we better have a good answer. The only way I know for giving a correct answer in this kind of situation is by having the answer written somewhere.

There are other reasons to try and know why a user is associated to a role, and one of them is "removability"; if the reason for a user to have a role is no longer relevant, it can be removed. If the role was given because of a business rule, if the business rule is no longer matched (e.g. the user no longer belongs to the department that "grants" the role), it should be removed. Recertification of roles that have been manually given is much easier than adding to the recertification the roles that have been given by automatic processes (of which the recertification may have no knowledge about).

9.1.1.5 Temporary user-role association

The relationship between a user and a role may be defined as temporary (with an expiry date).

One of the reasons for this is the tendency of human being to forget stuff that does not seem to be a priority, and (please believe me) removing privileges that are no longer needed is not a priority for most human beings.

I have heard colleagues arguing that all user-role associations should have an expiry date. I can see the point and I do not have strong arguments against the idea, but I feel reasonably comfortable with non-expiring privileges.

9.1.1.6 Multiple user-role relationships

A user must be able to have multiple relationships with the same role, because these relationships may have differences (even when the entitlements that they provide to the user may all be the same).

To illustrate this need, let's check this case:

- Adam belongs to consultancy department
- Adam is provided a role (let's call it "r_sales") that enables him to edit sales proposals for six months (enough time to help with a certain proposal for a customer)
- Adam is moved to the sales department, which grants him "r_sales" role
- Adam leaves the sales department, his department no longer grants him "r_sales" role

There are some conditions that have to happen:

- Adam has to have "r_sales" for six months
- Adam has to have "r_sales" while he is in the sales department

Apart from some other (more complex) ways of satisfying both conditions independently of the time when Adam joins and leaves sales department, it is possible to have a relationship Adam-6months-r_sales for six months independently of the relationship Adam-sales_department-r_sales.

9.1.1.7 Flexible role-entitlement relationship

The relationship between a role and the set of entitlements it provides must be based on parameters.

It may seem obvious that system administrators of google cloud and system administrators of amazon web services should have different entitlements. It may seem obvious that office employees in Mexico should have read access on customer information of Mexican customers and office employees in Spain should have read access on customer information of Spanish customers.

We may define different roles for different contexts (in addition to defining different entitlements), or we may push some intelligence in the role definition and let the information in the relationship user-role to address that matter.

The first option creates complexity. The kind of complexity in identity management; the one that grows exponentially (multiply the number of clouds by the number of entitlements, or the number of entitlements by the number of countries). The one we do not want to treat with.

Remember we also should have many user-role relationship and attributes in the relationship itself.

9.1.1.8 Relationship between roles

There are many different kinds of role to role relationships:

- **Inclusion / Inherits from**: If role A inherits from role B, role A provides at least the same amount of privileges as B. In addition, the changes in the set of privileges provided by B are automatically applied to A.
- **Exclusion**: If role A is exclusive to role B, if a user is associated to role A, it cannot be associated to role B. Please note this relationship should not be exclusive to roles. It should be possible to define exclusion relationship between roles, entitlements and systems (role-role, entitlement-entitlement, system-system, role-entitlement, role-system, entitlement-system)
- **Equivalent**: If role A is equivalent to role B, it provides the same set of entitlements. Equivalent roles are good candidates to be removed from the authorisation model.
- **Enables**: If role A enables role B, association of user to role B is either impossible or has no effect unless user is also associated to role A. This may be useful for example to separate management of privileges in a system from the ability to log in that system.
- **Other (custom) relationshps** (e.g. "creates risk when in conjunction with"): relationship that may be treated by business rules or live like annotations in the system. It may affect relationships between roles, entitlements and systems.

9.1.2 A deeper look into ABAC

So, we have discussed a bit about roles and role based access control. It may seem enough for most, but it falls short (really short) for many cases. Before showing some examples, let's talk about attributes.

Attribute (according to Wikipedia) is a specification that defines a property. One may think about it as a value (stored in some repository), but that is just one of the possible implementation. It may be calculated in real time, cached, etc. It is a definition, that may be implemented in an infinite number of ways. That said, storing a value in the database will suit 90% of the use cases.

We know what is an attribute, but what things may have attributes? Any entity; that is where the qualitative difference with roles lies. Attributes may be associated to the user that is making an operation, to the object that the user tries to access, to the roles the user is associated, to the environment where the operation takes place (e.g. time of day), to other entities affected by the operation (e.g. when a user operates between two bank accounts, we may wish to check her relationship with the set of owners of both accounts).

With this understanding about attributes, I think it is easier to think about scenarios where RBAC is not enough:

- A user can only access an application during working hours
- Employees with a range of "director" may see and comment documents classified as "strategic" and "Draft".

If RBAC is properly implemented (as discussed earlier), much of the requirements related to ABAC is already in place.

9.1.2.1 Identity dimensions applied to ABAC

In a RBAC authorisation model, and given authorisation is a matter of security, roles and role management usually fit under the umbrella of security. In ABAC attributes are introduced into the equation, and in some cases the management of these attributes is not done by the same team as the one managing roles.

It is important to understand the reliability of the information we are using. This does not mean that only information managed by the security team may be used, or that there must be a validation of the information before using it. It is just a call to notice the three dimensions of the identities: completitude, reliability and fragmentation.

If the information is not "complete enough", it is difficult to set authorisation policies that make use of fields that may be missing.

If the information is not "reliable enough", it is difficult to use that information for security matters.

If the information is too fragmented, it may be impracticable to evaluate attributes and authorisation policies with the performance required.

9.1.3 Is there room for MAC/DAC/ACL?

Yes. It may sound strange, after explaining RBAC and ABAC and explaining how these models cover extremely complex scenarios, and of course I am not going to argue that all authorisation should rely on a management done per-identity, but in some cases it is better and easier to do so.

If you have a pet, maybe you have to leave it to someone you trusted. In that case, either you bring your pet to that person's house, or you bring that person the keys to your home. In either case you are empowering one specific person because you trust her. That trust is affected by many factors, but most probably you would not trust any other person, even if that person has the same age, height, weight, professional experience, etc.

Attributes (and roles are just one type of attributes) are not all there is about an identity. We already discussed identity management requires accepting we are making tradeoffs and simplifications about identities. It is simply impossible to have *all* the knowledge about a person in a system.

Of course the use of an specific assignment of one user to an operation over an object (or any other form of DAC "policy") must retail the properties we have discussed: the action must be audited, the reason for the assignment must be recorded, depending on the criticality of the action it may not be allowed, etc.

Robert C. Martin in his book Clean Architecture shows that software development paradigms (structured programming, object oriented programming, etc) are just restrictions to the original programming methods with direct access to hardware resources. In the same way I think authorisation models are restrictions over primitive identity management mechanisms, well deserved and well needed restrictions, but restrictions none the less.

9.2 The reason for a complex authorisation model

When reading these explanations about an authorisation model, it may seem obvious to ask "Why should we take so much work into defining such a complex thing?", or "Why should we define the authorisation model as something so complex?".

There are two reasons for this:

- Complexity in the authorisation model definition reduces complexity in the identity management
- The authorisation model for a company is the framework for the authorisation models of the systems run in the company

The first statement is something we have seen many times, that a simpler authorisation model than needed may do the job, but at a cost of increasing the complexity exponentially. To illustrate the second statement, let's think about this scenario where a fictional system and a fictional company: the company uses an ACL for identity management and the system is based on groups (RBAC). It is possible to maintain an ACL with the association of users and resources, but the complexity grows exponentially.

9.3 Propagation of the changes in the authorisation model and the attributes

In order to be able to make an appropriate security enforcement of the authorisation model, it is required to have updated information. Information propagation should happen online; if not immediately, at least as a best-effort.

It sounds simple, but when designing an implementation for the authorisation model, it is necessary to take into account what elements of information are going to be updated and when are these changes going to be taken into account.

Chapter 10

The use of an identity

We know what is an identity, but in order to expand its meaning and its qualities, it is imperative to understand what are we going to do with it.

Even when the actual objective of the work with identities belongs to the worker, and this document cannot cover them all, let's classify operations in five different categories:

- Obtaining identity information
- Identity information storing and making available
- Identity lifecycle management
- Identity information improvement
- Identity information retrieval for actual company uses

Only one of the five categories is directly related to getting value from the identity information we have. May be that is one of the reasons why it is sometimes so difficult to use the information when it is needed: because it was not sufficiently elicitated, because it is not properly stored or not properly available, because it had not been updated, or because the information is too raw to make use of it.

10.1 Identity management processes

Here we will expand each of the identity use categories and provide examples. This section should also help making it clear why concerns are so different in each of the categories.

10.1.1 Obtaining identity information

Elicitation of identity information from sources external to the company. There are different ways of getting identity information. The following are just a few:

- **User self-registration**: Enabling users to register into our services
- **"Bankization"**: User goes through the classical process of going into a bank office, providing ID, etc in order to become a customer
- **Online registration**: Similar to the previous "bankization" process, but happening online
- **Prospect database**: In many cases a company buys a prospect database: a set of possible customers with some specific characteristics that make them prone to buying some company services or products
- **User behavior collection**: While a user uses our services there is much information we may gather (of course while observing regulations). This behaviour has two components: business information (which is the set of services that the user uses most, etc), and technical information (source IP address, used web browser, typing cadence, etc).
- **Merges**: During company merges, there is an addition to the set of identities managed by the system. This includes employees, customers and all related information.

All the information that comes into the company must be stored and made available.

10.1.2 Identity information storing and making available

Identity information is quite heterogeneous from a storage POV. The requirements for reading, updating and navigating identity information are extremely different depending on the use they are going to have.

From an IT architecture perspective, we probably want to expose identities in a versatile way. Instead of providing access to the repositories via ODBC, LDAP or some other repository specific interface, we may prefer a wrapping for most commonly used identity operations. Even if we do not plan to change the repository technology itself (LDAP looks like a long term option for identity storage), it allows changing the internal structure (which is something that will surely happen at some point). SCIM2 may be a good starting point for this wrapping interface.

10.1.3 Identity lifecycle management

Lifecycle management is not as simple as implementing a CRUD interface around an identity repository. It goes much further than that.

Since identity information is subject to many regulations (e.g. GDPR), and it is used throughout the whole organisation, the lifecycle management is very complex. It is so complex I would divide it in two layers:

- Lifecycle management of the identity repositories
- Lifecycle management of the identities in each repository

In order to have come control (centralised or not) over identity information, it is mandatory to know where are identities stored in the company. This seems to be easy, but it goes into the lands of shadow IT, department databases, application database exports, etc.

In this case I honestly think the typical IAM initiatives (the ones related to employee and contractor identities) have done a good job over the years and there is usually a good census about corporate applications and repositories for these identities and their relationships.

Regarding customer information and other identities, since there is no equivalent to the IDM (Identity Manager), I am not sure there is a good centralised census.

A complex part of the lifecycle management regards the authorisation processes involved in the evolution of the privileges of a user. The modifications in the authorisation model, along with the requests, approvals, auditing, etc. These processes are usually high level business processes that involve a great number of actors; its definition and implementation is not easy or fast at all. I would suggest to begin easy and go on improving the system.

Another complex lifecycle management operations is the recertification process (which has several implications regarding authorisation). In this process the company confirms that the employees and contractors have the right set of privileges to perform their job. This labor not only requires the analysis of the individual identities for employees and contractors, but also an analysis of the privileges provided by automatic means.

10.1.4 Identity information improvement

This set of processes are composed by the analysis an enrichment of identity information.

Different final users of the identity information will need different types of identity analysis and improvements. For fraud analysis, the kind of enrichment of information will be very different than for authorisation.

I will show three different scenarios for identity enhancement that will provide higher value to three different uses of the identity information.

- **Connections between known fraudsters**: In order to have improved detection of fraudsters, we may want to analyse relationships between know identities associated to fraud. We may think that two persons living on the same address may both be fraudsters.
- **Geolocation of a customer**: After having received user connections for a long period of time, we may decide that any country from where the user has connected at least 5 times is a "usual country" and hence we should not request a second factor of authentication when the user connects from these countries.
- **Authorisation model improvement**: At some point we may decide that we have too many roles in the company and we want to have fewer to ease the administration. This involves an analysis of the current identity information in order to come out with a better arrangement for the roles and permissions.

10.1.5 Identity information retrieval for actual company uses

This is the category where most eyes are set upon: extracting actual value from the identity information.

In this category we have the common uses of identities, such as marketing campaigns and authentication and authorisation of user actions, but we also have some less usual processes, like providing information about an identity to the owner of such identity, sending identity information to a third party to receive some valuable product or service (e.g. credit card printing), auditing identity information to prove compliance with regulatory constraints.

It is this final use of the identity information the main element to guide the decisions in the identity information storing and making available processes.

10.2 IDM for employees, contractors, consumers and customers

Now that we know a little bit about identity processes, let's have a look at the two common tools that help in the management.

10.2.1 The use of an Identity Manager (IDM)

Many of the requirements regarding identity information management involve synchronisation of information across multiple sources. We have already discussed the concept of identity federation, and how it goes beyond authentication and single sign on.

Unfortunately, identity federation cannot easily solve most of the problems associated to multiple identity processes (please remember, there is no silver bullet), specially when there is a complex authorisation model in place.

For cases where identity information has to be shared among different repositories, and there must be some kind of "higher order" identity, that contains information about authorisation that transcends any single repository in the company, that's the place for an IDM.

IDMs are commonly used to store high level information about identities, along with enterprise roles and some other authorisation information. In most cases, before using this identity information, it has to travel to a different system, where the identity information is consumed locally. This travel is called provisioning. The travel in the other way around is the reconciliation.

As we have already discussed provisioning and reconciliation along with auditing while speaking about centralised control; I am introducing the IDM here because it is the processes about identity that explains its need.

10.2.1.1 IDM as an application authorisation repository

IDMs provide a complex authorisation model, which should cover most of the authorisation scenarios in a company. This ability makes the IDM itself an interesting option to store and manage the entitlements local for an application. As we are talking about complex authorisation scenarios, this kind of system regards specially to employees and contractors.

It is not very un common, but it is a model worth exploring to use the IDM itself as a replacement for a local identity database or a local identity directory.

It provides (in addition to the authorisation model) a pre-provision of all the identities in the company, an immediate update of much of the identity information, etc. On the other hand, managing one application in such a complex component as an IDM may be overkill in many cases.

10.2.2 CIAM identity manager

If IDM relates to employees and contractors, CIAM IDMs relates to consumer and customer identities.

Most of the identity processes are different in internal IAM and CIAM, but both share much of the security-related use of the information: user profiling, authentication, authorisation and access control.

In the case of CIAM there is no interest (or not much) on using a complex authorisation model, there is on the contrary a huge attention to the consent management, as well as federation of identities with external sources (e.g. social networks) and analytics (from a security point of view, but also from a marketing point of view).

10.2.3 Convergence of identity management for internal and external users

Ambitious as this personal vision may be, I still cannot conceive in the short term any commercial tools that address both customers and employees use cases in an integrated manner.

But the need is already there; if we want to have a truly global and unified sight of all the identities that are related to the company, it is mandatory to integrate the information for IAM and CIAM (without overlooking devices, accounts and the rest of identities that are involved).

Chapter 11

Uses of an IDM

Identity Managers (IDMs) undertake to do the following things: Identity Governance, administration, provisioning and reconciliation. These operations make sense specially in the management of the identities over which the company has (or should have) tight control: employees, contractors, service accounts, devices, things, etc.

I am referring to IDM as a system that performs (or helps to perform) these actions. It is the same to me whether it is an ad hoc solution, a COTS or a hybrid in between.

11.1 What are these tasks

These are the most typical operations that the common IDM undertakes. Prior to explaining how they should be done, let's talk about what they are (just to avoid misunderstandings later on).

11.1.1 Identity Governance

According to Wikipedia, "Governance is all of the processes of governing", which may not be extremely useful; it also defines it as "the processes of interaction and decision-making among the actors involved in a collective problem that lead to the creation, reinforcement, or reproduction of social norms and institutions".

Interaction and decision-making seem to be a good explanation of the tasks involved, but in order to decide, one must know and understand, and the limits of knowing and understanding are stated by human brain capacity. What happens when governance is done without knowing and understanding is well depicted by (among others) The Simpsons.

Since governance requires knowing and understanding, I'll include in this category all the associated tasks, such as reporting, analysis and so on.

Let's have a session of truisms. The government (specially the decision making) is made by the governor, the governor must be a person and hence the decisions are made by people. Automating actions based on business rules does not remove the decision making from the people; the decisions are made ahead of time and actions are taken at a later point. "Delegating" decision making into an algorithm (from Naïve Bayes to so sophisticated AI) does not either remove the decision making (and the responsibility) from the people.

11.1.2 Identity administration

This refers to the CRUD operations on identities; specially referred to the operations on higher order identities (the ones usually stored in the IDM). The operations on these higher order identities usually originate operations in the identities (generally system accounts) in the IT systems attached to the IDM.

11.1.2.1 ¿Who should perform the administration of the identities in the IDM?

Identity changes originate by decisions of people: a person switches to a different department, comes to the company, leaves the company, requires to use a certain application to perform some beneficial process for the company, etc.

It is the people making the decisions who should "operate" the IDM. It is the people making the decisions who should state those decisions and use the tooling, interfaces, etc in order to do so.

Identity administration should be performed by the people who require the modifications into the identities.

Of course this rises two important questions: ¿How can "business people" without technical knowledge about identities take actions on them? and ¿What do

IDM administrators do? The answer is the same: IDM team must provide the tools for the "business people" to use "their language" when making modifications to the identities.

11.1.2.2 Other CRUD operations

Administration of the identities should be performed by the identities consciously making decisions that affect identities, but not all operations on identities can be related to a conscious action.

There are many situations that affect identities without a conscious action (but always with consent): elements of information collected for behaviour analysis (e.g. device used, source IP address), operations on bank accounts that will affect the credit score, re evaluation of information (e.g. recalculation of identity score), etc.

These operations that happen around the identities and are not triggered by a conscious decision, should be automated and the implementation of the automation should be done (or guided by) the identity team.

11.1.3 Identity provisioning

Identity provisioning is the act of pushing identity information from a central system (usually the IDM) into a provisioned (managed) system.

This requires that the IDM talks the language of the managed system, including understanding the semantics of the system authorisation model and the mapping to the IDM's. This is one of the reasons why the authorisation model of the IDM must include the one of every managed system, and hence be at least as complex as the union of the complexities of all the managed systems.

11.1.4 Reconciliation

It sounds more civilised than it usually is. This process matches the information in a managed system with the information in the IDM, and takes action on the discrepancies.

The options are these:

- Discrepancies are not detected (e.g. the password has changed, but the IDM does not care about the password field). In this case, the change is ignored.
- Discrepancies are not understood by the IDM (e.g. the differences happen because the user belongs to a new group, but the IDM does not understand the "group" concept of this system). In this case, the change is ignored.
- Discrepancies are detected and the IDM is defined to possess the truth. In this case, the information in the IT system is overwritten with the information in the IDM.
- Discrepancies are detected and the managed system is defined to possess the truth. In this case, the information in the IDM is overwritten with the information in the managed system.
- Discrepancies are detected, but there is no decision taken about which component has the truth. In this case, discrepancies are noted (in the IDM) and no action is taken on the managed system.

Even calling this document a vision, I would not dare to say that in every single situation the IDM will hold the truth and the IT system will not. That would deny the need for operative decisions; these decisions should be the exception, should only be allowed for certain situations (e.g. the address may be changed, but not information about critical privileges), for a brief period of time (because all relevant information managed by the IDM should be consistent between the systems and the IDM) and for good reasons (e.g. the attribute is not properly managed by the IDM).

Of course, one may think it is possible to have a single source of truth and radiate ultimate truth from that point to the rest of the systems. It would be a silver bullet, and I have already said I do not believe in them. I prefer to think on a system flexible enough to deal with reality.

11.2 Workflows

One important part of an IDM solution, and a certainly needed element for the empowering of "business users" into identity administration, is the implementation of tools that support the processes where identity modification happens after the agreement of multiple people: the dreaded "workflows".

Workflow implementations usually contain a mixture of the real business process that wants to be "implemented" ("tooled" could be a better word), some inter-personal relationships that do not belong to the official process but happen nonetheless and useful resources that serve the people involved in the process. It also contains many other things I am not going to get into: steps for information addition, steps for possible delegations (in case one person or group cannot participate for some time), steps for escalation (in case one person or group have unexpectedly not participated in the process for some time), orchestration of technical processes, waits for events to happen, etc. All these components are useful and in many cases necessary, but for the sake of simplicity I'll think about them as a sequence of approvals necessary for an action to be taken on one or many identities.

11.2.1 The basis for identity administration

As explained before, identity administration should happen as a conscious operation of a person taking a decision, and the means for that decision to be processed may be thought as a workflow. The case of a direct edition of an identity record to update the name of the user may be thought as a single step workflow.

This approach may be overkill for most technical operations, but it comes handy to establishing a common ground for all operations. And remember, this is about a vision; it does not need to be close to reality ;)

One important part of treating all administrative operations in the same way (even those actions performed automatically) is the ability to store the reason for the action for every action. This way, it is possible to understand why does a user have certain privileges, why an account was created, etc.

11.2.2 The accountability for non-personal identities

We have discussed that there are many identities that do not relate at all with persons, meaning that they are not the avatar of a person. Examples of these kind of identities include devices and things, but there is a case that is mostly overlooked in conventional IAM projects: service accounts; identities that identify systems or services.

Services cannot (should not) be managed in the same way as personal accounts. A service is not going to decide to change its own password, it does not check its own information to ensure it is up do date, it does not have right to be forgotten (even when they are forgotten many times), etc.

In order to make proper management of the service accounts, it is necessary to identify the entity they refer to, create as many identities as needed (e.g. an application may need a service account to access an LDAP service and another account to access a database), associate them to people (preferably more than one person, just in case…), make those people accountable, etc.

The processes involved in the management of these identities are perfect candidates for workflows (even when the people needing these identities are usually very technical people), because that way operations are accounted, traced, approved, etc.

In many cases there should be no person knowing the secrets (e.g. password or private key) of a system identity, and this can "easily" be achieved by wrapping administrative operations with workflows.

11.2.3 User privilege cloning

When someone (let's call him Jake) joins the team and is going to do "more or less" the same job someone else (let's say John) is already doing, there is a very typical request: "Give Jake the same privileges as John". This may happen for just one system or for all of the systems of the company.

This has been demonised over the years as a bad practice because it is necessary to understand that the privileges being granted are bering grated for a reason. When cloning the privileges of a user into another one, there is no acknowledgement that these (and not less) are the minimum required privileges for the newcomer.

In order to circumvent this, I have heard managers asking for an exhaustive list of the privileges of John just to be able to make a proper request with all the privileges that Jake should have. In this case, the reason for the privileges of Jake will be listed as "requested by the manager", instead of stating that it is a "user cloning requested by the manager".

The only option I can envision to prevent this is by allowing user cloning. This enables labelling these privileges as "cloned from John", and that labelling enables (even when does not guarantee) a proper management when John changes his privilege set.

I prefer to adjust my model to reality instead of trying to make reality adjust to my model; mainly because reality is much smarter than me.

11.3 Complexity of the authorisation model

In order to provision and reconcile a system, it is necessary to understand that system. In the context of an IDM, this is generally related to authentication and authorisation, so it is also related to the authorisation model of the system.

When you have to manage (i.e. understand) many IT systems, it is necessary to understand them, including their authorisation models. This makes it necessary to include all the authorisation models of the managed system as part of the IDM's model.

In the context of identity management, mistakes are paid in exponential complexity. It is [more or less] obvious that it is much more complex to manage user-resource permissions instead of user-group and group-resource permissions. If an IDM could only manage user-resource permissions, and a managed system would enable groups, the IDM connector (the piece in charge of the provisioning and the reconciliation) would have a really difficult time mapping information to and fro the IDM, the IDM administrators would have a hard time understanding the implications of the actions, etc.

If the IDM is not [trivially] capable of managing the authorisation model of a managed system, it is going to cost dearly to properly manage (specially understand) that system.

11.4 Where entitlements belong

Entitlements (the privileges that enable operations on resources) are obviously related to the managed systems. An entitlement in a system does not mean much (¿anything?) in another system.

Entitlements for a particular system may be stored inside the IDM or in a different repository, and when authorisation takes place, that information may be retrieved from the IDM or from that other different repository. Let's analyse the options.

11.4.1 There is no IDM; only the system repository

If there is no IDM, there is no provisioning or reconciliation; there is no centralised knowledge about the identities in this system. We can call it an outlaw, but it is a common situation in many systems (e.g. administrative accounts of the laptops of contractors).

Anyway, if there is no IDM, it is not a discussion for this section.

11.4.2 IDM as sole repository for authorisation

In this case, it is the IDM the only one that stores entitlement information. There is no "different repository".

IDMs are very well suited for storing and managing authorisation information, and are a great place to store some other information that may come handy.

That other information is a downward slope. Storing first and last name of the identity may be a good thing: it helps understanding the identity and it also can be useful to be propagated to some external repositories (or consumed by external systems), but the line between storing all the information in the world and storing just the useful information is a matter on how down the slope to chaos you want to go. The more information you store, the more difficult it is to change, evolve and please everyone relying on the system.

Regarding authorisation information, it is very important to understand that using the IDM as the sole source for authorisation for a system makes the IDM itself part of that system at execution time: if the system needs to be backed up, the IDM needs to be backed up at the same time (or at least maintaining coherence with the rest of the information of the system), if the system needs lightning fast authorisation, the IDM must be lightning fast … for this system (while servicing other systems), if the system needs to run in different countries, with different policies, the IDM must follow through, etc.

This scenario remains in my vision because there are many applications that would benefit from this approach, and many of them will also benefit from externalised authorisation.

11.4.3 IDM plus another repository

So we have decided to duplicate information, this is not the option for the faint-hearted, as there are some things to consider…

11.4.3.1 Entitlement Synchronisation

We have information in two places; we better make sure it is the same. This calls for synchronisation (at the entitlement level; not just at the account / account level), which may happen periodically, on demand or can be triggered by changes in each of the ends.

Of course this is not easy, but as this is mandatory for the rest of the discussion, let's think it is a properly solved problem and the information is synchronised.

11.4.3.2 Information to share

We can decide to share all the information or just some of it. If we choose to keep secrets from the IDM, we better have some good reasons for it.

The IDM can only manage the information it knows about (another truism). If we choose to hide "low level" entitlements and decide to share just "relevant" entitlements, chances are at some point we will forget to share a "relevant" entitlement. In addition to that, it is important to understand that governance processes (and audit, etc) will only be as effective as the information shared allows.

On the other hand, we may decide to store all the entitlements in the IDM. In this case, we better have a proper authorisation model in place in the IDM, but also in the managed system.

11.4.3.3 Bringing the complexity into the IDM

Remember that adding information from different sources adds on to the complexity of the IDM.

If the managed system manages user permissions for a branch office, and we define 100 different possible roles in an office and we have 1000 offices, we have two options: either the system enables the definition of entitlements based on the actual office and the actual role, or we create all possible roles for all possible offices.

In the first case we would be needing around 1100 elements, in the second case, we would be needing 100000; the difference between addition and multiplication.

Please note we may have the best, most complete, most complex authorisation model in the IDM, and even in that case, if the model of the managed system makes us create 100k roles, and we choose to share that information with the IDM, we will have 100k roles in the IDM for one single system.

Once we have added all the information from the managed systems, we may want to have additional roles, compositions of these roles with other roles, etc.

Remember: in identity management mistakes are paid in exponential complexity.

11.5 Identity management as an agility enabler

Identity management is one of those disciplines regarded as business stoppers for a long time, just like some kind of tax with no visible ROI. This same thing has happened with security for a long time.

In the case of security Sony, Ashley Madison, JP Morgan and many more companies have shown that security is necessary, and something that should be sought after better sooner rather than later. So far (fortunately) there are not many companies that have shown the world what a wrong approach to identity management implies. This is because a bad identity management strategy is a silent killer.

We have discussed many times that mistakes in IAM are paid in exponential complexity. This means that the administration of the overall system is never impossible. It simply takes more and more resources, but it is very difficult to draw a line on the sand and say "enough".

A good identity strategy enables the agile principles; empowers knowledge workers and accelerates innovation.

The consequences of a bad (or at least not good enough) identity strategy has two "smells":

- It is very difficult to make something work; specially in the beginning or when relevant changes come into place. There are many reasons for this; for example: identity management processes focused on auditing instead of enabling, manual steps (and authorisations), and many other situations that have to be tackled manually because they could not be thought in advance.

- It blew up at some point and you get to know it too late. This moment usually comes at the realisation that the company has twice as many roles as it has users, that the number of business rules is incommensurable, that the discrepancies found during a reconciliation could be less if you had tried to reconcile a system of a different company, etc.

It is extremely difficult to design a good identity strategy (I hope this document can shed some light on what to take into account), but its consequences are great. Imagine a world like this:

- When a newcomer has all the privileges he/she needs the day he/she comes into the company (including desk, laptop, mobile and a welcome pack with a t-shirt of the company, some sweets and a handwritten letter of welcome. I acknowledge I have received such a welcome pack.
- When a person changes department, privileges change accordingly.
- When a person leaves the company, all privileges are revoked.
- Most of the time, things simply work and people can do whatever they need to do.
- When someone needing a privilege knows what to do: how to know what is needed, how to request it, the process that the request is going to follow (if any), who is responsible for each step, the implications of the request (e.g. whether it is risky for the company), etc.
- When someone needs to be involved in an identity management process (e.g. approval of some privileges change, or a recertification), that person knows his participation is needed, why it is relevant, why it has to be done by him/her, what to do, how to do it, etc.
- Compliance is guaranteed and easy to prove.
- Policies are automatically enforced, violations are addressed according to its relevance: critical transgressions are not permitted; trivial matters are notified and tracked.

If there is a word for identity strategy definition it would be “preventive”. In order to become a true enabler, identity management must focus on anticipating the needs of the company, specially because solving existing problems tends to create “patches”, and these patches will possibly not destroy the system; they will (I’m sure at this point you already know how this sentence ends) create exponential complexity.

Chapter 12

Authentication, authorisation and access control

Regarding security, the most common uses for identities (other than information improvement, governance, auditing and the like) are authentication, authorisation and access control.

In order to perform these actions appropriately, identity information must be fresh and accessible. We have already discussed where this information should be and how to maintain it up-to-date, so we will not discuss much about these matters.

Chapter 13

Authentication

I'll try to depict the options for a good authentication schema for identities whether or not they are people.

This discussion is oblivious of the protocols used for the authentication.

13.1 Classification

Authentication as the process of verifying an identity may happen in many forms. In order to make a simple discussion, let's make an almost classical grouping:

- **Something you know**: some kind of secret that no one can guess as it is within the internal memory of the entity.
- **Something you are**: something so intrinsically belonging to the entity that no one else can be.
- **Something you have**: some element that is tied or associated to the entity and cannot be stolen.
- **How you behave**: some patterns in the observable behaviour of the entity that cannot be reproduced. In this category I refer to the human-computer interaction as well as the higher level behaviour related to the, order of actions inside a web site, common business operations, etc.

13.2 Extending the categories beyond people

In most cases this classification is associated to people, but they can also be associated to other kinds of entities that have to be authenticated, such as devices and services. I'd like to analyse other alternatives which enable a better authentication schema.

A part of defining that authentication schema is to properly think of the authentication methods as part of the authentication strategy, instead of side efforts to improve security.

13.2.1 Something the service knows

This is the most common way of authenticating a service, by something only the service knows. This element of information may be something in a configuration file or inside the source code or a mix of both that seems to provide a higher level or security.

The secret may be presented as a bearer token, exchanged for an access token or used to create a cryptographic token. Anyway, the secret is used as the basis for the authentication process.

13.2.2 Something the service is

It is not obvious to think of something that a service "is". It does not have biometric unique qualities; at least it does not have the "bio-" part of these qualities.

But a service is unique. There is a very specific code that the service runs and that generates the answers that the caller seeks. This code can be hashed and that hash is unique.

Having a code that hashes itself may be complicated, but nowadays services rarely run on their own. We have application servers, virtual machines, container orchestrators and many other components that host the services. These hosts are very appropriated to hash the services.

That hash (possibly signed) is a metric of the identity of the service. If "biometric" is not a suitable word for this, maybe "inert-metric" could fit.

13.2.3 Something the service has

If it is difficult to think about a service as something that may have biometric characteristics, thinking about it as something with pockets may be even more difficult.

The thing is that this has happened in the past as something quite common. I'm referring to the "backpack" that had to be attached to a serial or parallel port (yes I'm old and proud of it) in order for some software to work. If the software was installed, but the device was not attached to the right port, the software did not work.

That was a "something you have" authentication. And that same approach can be done too for IoT devices (nowadays it would be done via USB).

The contents in the attached item has something necessary for the correct authentication of the service. It may be some secret that is validated, some autonomous system that authenticates itself or some part of the code of the service that runs exclusively inside the item.

Another type of "something the service has" option is the use of an HSM. In this case, the service cannot authenticate without the access to the HSM. This access may happen as a local hardware associated to the machine the service runs on, or a network HSM (which carries its own authentication complications).

13.2.4 How the service behaves

The behaviour of a service is something that can be defined, analysed, learnt and verified.

Of course we expect computer systems to behave as expected. That may be a double edged blade: it enables us to work with systems in a very trustworthy way, but it also makes us "lazy" regarding behaviour verification.

One example of learning the behaviour of a system is done by many WAF (Web Application Firewall) vendors as well as database protection solutions. In both cases, the system observes inputs and outputs and defines a behaviour based on many parameters: set of characters used, number of rows returned per query, set of database operations, set of http endpoints and verbs, etc.

This information composes the expected behaviour of a system. Violating this behaviour makes this kind of authentication to fail.

13.3 Authentication schema

An authentication schema is a definition of the possible authentication scenarios that are required in order to ensure that the authentication risk is low enough for the operations being requested.

These operations can be as broad as "application access", which would encompass any operation in an application, as narrow as "this specific operation", which would be valid for just a certain operation, or something in between like "financial operations of up to 20 Euros (with a sum per day of 100 Euros)".

13.3.1 Selecting authentication mechanisms

Now that we know that humans and non-human identities can authenticate in many means, we have the playground to define a rich authentication schema.

In a communication, there are at least two entities that should be authenticated: sender and receiver (e.g. user and system). We may have more components, such as the mobile device, the web browser, the infrastructure where the service is deployed, etc. The security should be secure for all involved parties, and all components should be authenticated.

13.3.2 Defining trustworthiness of each mechanism

Once we have defined the set of agents involved, we have to define the authentication mechanisms available for each of them. Once that set is defined, we have to choose which mechanisms are "valid" for our standards; a subtler categorisation would be to define a trustworthiness level.

For example: when we decide to use a user's source IP address during the authentication process, we should classify it as an authentication mechanism, define how trustworthy this information is and how much it impacts in the risk level of this authentication. Same goes for any other element of information for all the involved parties.

In opposition to the authentication mechanisms (that rise the trustworthiness of the authentication) there is the time. As time goes by, the trust in the authentication process decays. This decay may be immediate (authentication is valid for just one operation), have a defined expiry date (e.g. classical web sessions) or decay over time (e.g. with a formula that rises the risk level depending on the time since last operation or las authentication).

13.3.3 Defining operation sets

Not all operations are created equal. We may decide that a corporative application for the employees to see their payrolls has much higher authentication requirements than another one to approve expenses. If we want to think about corporate applications as one single application with many subsets of operations, we have come to the operation sets.

In some cases we may decide to define specific operations that require specific authentications, such as financial operations signing.

13.3.4 Classifying operation sets

No matter how we group operations, after the grouping we have to define a risk level associated to them. This risk level must be consistent with the levels defined for the authentications of the components involved in the communication.

For example: if we decide to create a set of operations called "mortgage read" (that binds together all the operations about present and past mortgages of the customer that do not modify them), and we decide that the user has to be authenticated "level 4", and the device must be authenticated "level 0", and the server must be authenticated "level 3", it is mandatory to define those levels and how to achieve them with a combination of the authentication mechanisms defined for the user, the device and the service.

13.3.5 Continuous authentication

Some of the authentication mechanisms can (and possibly should) be used throughout the communication.

Some of these actions include the behaviour of the user, evaluating the interaction with the device in every step, some others include the use of the same keys in the SSL connection. The first is usually regarded as a continuous authentication mechanism and the second one is just taken for granted, but it is important to think about it as a continuous authentication mechanism, part of the authentication schema.

13.3.6 Environmental factors

In addition to the specific authentication components, there are other factors that may affect the authentication processes.

These factors include the coincidence of certain patterns that have been associated to threads, such as the use of a vulnerable version of a browser, patterns in the cadence of interactions with the service, etc.

These environmental factors cover anything that may generate "suspicions" in the authentication, but are not strong enough as not to grant authentication.

The reaction to these factors may be as mild as to change the interaction flow (to discard the use of a bot), to request additional authentication (based on the rest of the authentication schema), to even reject access.

13.3.7 Business behaviour

A behaviour authentication method that may be too soft to be classified as a mechanism is the business behaviour.

The business behaviour is the observed behaviour of a user at business level.

This includes the set of usual bank accounts the user transfers money to, the sequence of links the user clicks before deciding to make a transfer, the type of items the user buys, etc.

This behaviour is relevant to estimate the risk of the operation, and hence is relevant to decide the authentication level that should be requested from the user.

13.3.8 Authenticating services in a microservice architecture

Once we have defined an authentication schema that includes authentication of services, implementing proper authentication in a microservice architecture is much more straightforward.

All service connection should require mutual authentication, encryption of the communications and possibly the sending of additional identities that may be relevant for the operation (such as the identity of the user performing the operation, and possibly the list of services that have been involved in the request).

13.4 SSO (Single Sign On) as a flexible concept

The way that SSO is understood in most of the security literature is based on the use of some kind of token that enables the user access to different systems, all of which are adapted to that token. Possibly the most commonly used SSO schema is Kerberos.

From my point of view, SSO is a perception of the user. If there is some means to make the user sign in just once, we are into SSO. Of course security level must be maintained.

The difference in the approach resides in the set of tools to make SSO possible. In my approach, it is possible to use desktop applications to detect login screens and inject user credentials, transparently translate tokens of one type into tokens of a different type in order to accommodate systems that can be integrated via different SSO solutions.

Let's think about a user that wants to sign in once and use the following set of systems:

- A system that understands only Kerberos
- A system that can be integrated only via SAML
- A system that does note understands anything else than OIDC
- A desktop applications with proprietary user databases
- Operating system credentials
- Pre-boot authentication components for drive encryption

Convincing all involved vendors to evolve into one specific SSO technology is not a viable approach; instead of going that way, we need an external system (possibly associated to the IDM) that can understand and manage accesses to all these types of systems.

This SSO system requires many components; among others:

- Integration with Operating Systems (please note the plural) credential managers
- Integration with the IDM or central user repository to know when a password changes and when a user is disabled
- Integration with Web Browsers (once again, the plural) to detect login screens in web pages (including technologies such as Flash)
- Integration with Operating Systems window managers to detect login screens in desktop applications
- Integration with command line applications to detect login requests
- Credential management to manage different credentials (including passwords) for different systems
- Policy engine to define access control policies for different systems; not all applications are created equal and not all authentication requirements must be the same
- Token manager to manage exchange tokens of different protocols (Kerberos, SAML, OIDC, etc)
- Authentication manager to manage different authentication mechanisms:
  - Secrets, specially passwords
  - Biometric devices: fingerprints, facial recognition, iris analysis, etc
  - Behaviour: including human-computer interaction as well as higher level behaviour such as the order in which corporate applications are opened
  - Physical tokens: Smart Cards, RFID, NFC, Bluetooth devices, etc

The security considerations of a system that enables access to to many systems in so many ways are paramount. It is obvious that the SSO system must be though, but it is very important to understand how wide it is. The attack surface is really big and all parts of the system should be thoroughly checked.

13.5 Biometric information storage

Storage of sensitive information is never a trivial matter, specially because this information (even encrypted) may be stolen. In the case of biometric information, it becomes a little bit trickier.

Storing information in a single place involves some risk. There have been enough data breaches about usernames and passwords as to understand that using the same password for a long time or for different systems is a risky practice.

One main difference between biometry and secrets is that you cannot change the first; this is both an advantage (you cannot "forget" the shape of your fingerprints) and a disadvantage: if your fingerprints are compromised (and you are leaving your fingerprints in a huge set of places every single day), it is difficult to "update your credentials".

Authentication information may be stored in a centralised server or in a distributed device. The first option may end up in huge data breaches of biometric information. The second one may become an issue for an specific user if the device is stolen.

Apart from all the security measures that can be applied to store information in trusted parts of the system, encryption of the information, and other measures, some part of the problem will remain: sensitive information is being stored in some place.

If this storage were not complication enough, sending biometric information through a network, even via an encrypted channel is forbidden by some policies. This makes it necessary to make a local processing of the biometric information prior to sending it (if at all).

Biometry is a complicated matter and it is very easy to do it wrong. Few standards have come to try and help interoperability and guide implementations. One of them is BOPS. Among other matters, it addresses the storage of biometric information split between the device and the server, mitigating the risk of information theft.

13.6 Authenticating a part of an identity

Even when it is not the most common scenario, there are cases where it is not necessary to have full identification of an identity in order to work with it (e.g. grant access to resources).

One of such cases is the buying of tobacco from an automatic machine. In this case, the machine does not need to know our identity, but it needs to be sure we are legally able to make the purchase. This case is solved in countries like Germany and Italy by presenting the national id card to the machine prior to using it as any other vending machine.

In these cases I always think about the following elements:

- An entity that does know the identity of the user: in the previous case, the government
- An identification token with the required information: the id card
- A mechanism to get the required information: the reader in the tobacco machine

In the case of the tobacco machine and the id card, the token contains much more information than the bare minimum to make the authentication, but we rely on the id reader to not track the rest of the information.

In some protocols such as SAML this scenario is well supported by using disposable identifiers and adding the set of claims that the service provider requires. In some other contexts, this may be a challenge.

13.7 Talk or share in delegated authentication

Authentication usually happens in a delegated way. There is a party that makes the authentication and another party that relies on the first one and accepts the identity.

There are two possible ways for this to happen: either these two parties share some cryptographic element of information (share) or there is a communication between them in order to verify the authentication token.

Each has its own pros and cons, but we can summarise this:

- When sharing, scaling can be difficult, as newcomers have to share information with at least one party, and that one party has to share information with all of them.

- When talking, there is always an overhead in the process due to the validation of the token that has to happen in the authentication party.

As always, there is no single approach that fits all cases. There is no silver bullet.

Chapter 14

Authorisation

Given we have already discussed (long and hard) about authorisation model, let's think about authorisation as a "simple" mean of applying that model.

14.1 Where authorisation takes place

There are many places where authorisation can take place. For a comprehensive analysis of these options, there is a good document from Gartner called Decision Point for Selecting Runtime Authorization Architecture Patterns.

In this document we will explore just enough to discuss EAM (Externalised Authorisation Management).

14.1.1 Within the service

Authorisation has classically happened within the application. It is some code inside the application the one responsible for checking that the user belongs to the admin group, that the resource belongs to the logged user and so on.

This is very convenient from the developer point of view, but haves two severe constraints that do not:

- It is not easy to analyse: it is necessary to go into the source code in order to understand the policy

- It is not easy to change: it is only the application developers who may make modifications to the policy

These two constrains (which are not the only ones, but IMHO are the most important ones) make the case for EAM.

14.1.2 Before the service

In this model, there is some component that intercepts the request to the service, makes the evaluation of the authorisation and forwards (if the evaluation grants it) the request to the service.

This model, as uses an specific component with independent governance from the service, removes the constraints in the previous section, but creates a new one: the result of the evaluation is not business related.

We may decide that the request should be processed or not processed at all, but it is not possible to take other actions without deep knowledge of the service business.

14.1.3 Next to the service

In this model, all requests go into the service without authorisation validation. It is the service the one that makes the request to the authorisation component and takes action on the response.

For example, a money transfer service may reject or accept the operation as a result of the authorisation, but it could also decide to mark the operation as "suspicious" or "risky" or delay the operation actual execution while informing it is done, or making any other business-related operation. These operations even when they may be related to the authorisation (which is a security concern), are business dependant, and can only be properly managed by business, not security.

The improvement of this model over the previous one is that the actions associated to the authorisation may be specific to the business of the service; the restriction that it generates is that deciding whether or not to call the authorisation module at all is a decision of the service.

14.1.4 Adding models external models

Of course it is possible to make two authorisation evaluations: one before the actual call to the service, and another one from within the service.

This approach solves some of the restrictions of the two models (the decision of calling a part of the authorisation is still dependant of the service), and may create a new one: an increment in the set of steps and hence cause an impact in performance.

14.2 The result of the authorisation

We have already talked about different kind of actions and security actions when discussing where does the authorisation take place.

To summarise and add to the list, we have the following options as a resulting action to the authorisarion evaluation:

- Security action: accept, reject, forward the request to the service, answer with some error code, etc
- Business action: actions that only the service can take and depend on the logic of the service
- Operation modifications: these actions are yet to be discussed, and include changes to the request that the service processes in order to be compliant with authorisation restrictions

14.2.1 Operation modifications

In some cases, a request made by a user must be narrowed in order to be processed.

An example of this is a request to remove everything sent to a cloud storage system. All of us expect the service to remove the information of this user, and not the information from the rest of the users of the system, even when the user has requested "everything".

The transformation from "everyting" to "everything owned by this user" is transparent to the user making the request, but is compulsory to implement it and to do it well.

If the authorisation is complicated (and we have already seen how complicated it may get), transforming requests is more complex. For example, if the set of elements that the user may delete includes the resources created by the user, plus the resources he's been given administrative rights to, plus the resources created under the hierarchy of resources created by him, etc. In such a case, the modification of the semantics is harder.

If done properly, the authorisation model should guide the modification of the requests, and these modifications could happen before the request actually comes into the service processing the request.

14.3 Getting the information for the aurhorisation

The information for the authorisation has to be gathered at some point in time.

This information may be gathered when the initial authentication process is made, when the first request is made or when the action is being processed.

In addition to the considerations of (Lazy evaluation)[https://en.wikipedia.org/wiki/Lazy_evaluation], the authorisation has two additional constraints that must be taken into account:

- The information should be as fresh as possible (with different freshness thresholds for different elements of information). If user information is gathered at login and it changes within a session lifespan, the authorisation process may take wrong decisions.
- The set of information required for an evaluation may be unknown. If the user logs into an application and the information of this user is gathered within that scope, if the underlying services call other services, chances are some authorisation information may be missing.

The first restriction may or may not be an issue depending on the duration of the sessions; if the session is too long, authorisation may be impacted.

The second restriction is a problem in microservice architectures. Either we have full knowledge at login time about the set of attributes that are going to be used throughout all requests, or we get "all the information" just in case. The first option is really complicated; the second option may have a huge performance impact since all requests will have a lot of information about the calling identity (or identities).

14.3.1 Two different approaches: OIDC (OpenID Connect) Vs XACML (eXtensible Access Control Markup Language)

I am not going to discuss protocols in this document, but it is important to note the differences in the approaches of these two protocols.

OIDC (as OAuth) is based on scopes, which can represent [more or less] anything related to claims and authorisation. These scopes are requested at login and are discussed with the service granting the tokens. As OIDC tokens can be self-contained, the token used to access can include the scopes.

XACML is based on evaluating authorisation by requesting identity information to external sources during the access control.

The first approach is great when the scope of the system is well defined and the governance can be done in a centralised way. The second approach enables distributed management of the authorisation and "Just in time" evaluation of the required elements of information.

My opinion is to combine both approaches, so general authorisation information is set at login, and information specific to a service is retrieved when it is needed.

14.4 Improving the time to evaluate authorisation

It is attributed to Einstein defining insanity as "Doing the same thing over and over again and expecting different results".

Getting the same elements of information for authorisation and evaluating them once and again may not be insanity, but will certainly result in a performance impact.

There are two main ways to improve the time for these evaluations:

- Cache the authorisation request
- Cache the identity attributes

14.4.1 Caching the authorisation request

The evaluation of two authorisation requests with the same values (including the applied policy set) should result in the same result. The question is "what does two authorisation requests the same?”

Of course, if all attributes involved in a certain request are the same, it should be obvious that the result should be the same. This is the easy one (or not?).

Of course, if the attributes that change are not evaluated in the policies, the result should be the same. This is easy too.

What happens if the change is subtle?

- If the change is the source IP address, it may imply a different geolocation, or the use of a round-robin in a gateway
- If the change is the timestamp of the request, a difference in just few milliseconds may result in changing the time when a market is closed… or may be not.

For a “similarity check” to be successful, it is required to understand the semantics of the business and the impact of each and every attribute change.

Let’s go back to the “all parameters are the same” case. It is often thought that actions are idempotent, but they are not in many cases. Being able to get a loan for one million euros does not mean being able to get a second (or a third) one. Once again, it is important to understand the business in order to cache authorisation requests.

14.4.2 Caching identity attributes

There are many questions relevant to cache an attribute: May it change? How often? How relevant is the attribute for the authorisation? How critical would it be if the attribute is not up-to-date?

It is required to balance the answer to all these questions in order to have a good caching strategy.

As it happens in the case of caching authorisations, it is mandatory to understand the business of the authorisation and the business of the service associated to the authorisation in order to make good decisions about caching.

14.4.3 Cache management

All information in the caches for authorisation should have an expiry date. Even when it is arguable in a theoretical way that some evaluations will never change, in most cases it is practical to spend some milliseconds (or even seconds) to make an evaluation instead of relying on old data.

The availability of the information in the cache is very difficult to achieve, as the information is diverse and the user may happen at different points. I would try a best-effort cache information sharing that could impact performance but not service availability. If an element of information (that has already been used) cannot be retrieved from the cache (either because it has not been propagated yet or because it has been deleted), it should be requested to the proper source.

Of course all cache management considerations such as refreshing expiry dates of accessed records, volatility of information, etc apply.

14.5 Carrying identity information

Microservice architecture are based on services calling each other. In this case, it is not the user directly calling the services and the set of components in the authentication is increased.

Let's suppose we have a proper authentication schema in place, an authorisation model we feel comfortable with, with all the means to get identity attributes and all the authentication mechanisms are set in place. Even in this case in a microservice architecture, there is a question with a non-trivial answer: "Who makes the request?".

When a user requests goes to a "perimetral" service it is easy to know who is making the request: it is the user making the request.

When that same request goes "bouncing" from a service to another, the answer may not be so straightforward: is it the user making the request or is it the service making it? Is the user's desire to make this call or is it something that we decide on his own? What set of privileges should be checked: the ones of the user or the ones of the invoking service?

Unlike other disjunctives discussed in this document, the carrying of identity information across a microservice architecture must be coherent throughout all the services. Hence it is a global decision that affects all the services.

On one side of the options is the simplest one: it is the service calling, so we remove all other identities from the request. On the other side is the most complicated: we stack the identities of all involved parties, including the requesting user (if any) plus the identities of all services involved.

Possibly the equilibrium involves the use of an arbitrary set of identities with some tagging for the role they play in the operation such as "initiator", "impersonated", etc.

14.6 The thin line between business and security

Authorisation is one of those concepts that are perfectly clear for people with different backgrounds. Unfortunately, their understanding is different and they usually do not know it. Authorisation from a business point of view relates to business logic; it has to do with account balance, solvency, etc. Authorisation from a security point of view relates to user profile, privileges, etc.

It is fun to speak with security people and say things like these: Authorisation is a matter of security; it is one of the things that maintains the company safe. For example, if we don't check the account balance before authorising a money retrieval, a customer could steal money from us. Hence, a security policy should be placed to prevent this from happening.

I know many security professionals who accept this set of statements without blinking. There are others who bring into context business policies, that may (or may not) grant a user to withdraw money as a loan; as a service that the company provides.

Checking the balance of an account and rejecting an operation when there is not enough money is a policy that could be regarded as security. Calculating a credit score seems closer to business. Evolving this policy as business requirements change is definitely something that cannot be done by security on their own.

Deciding which policies should be lead security is difficult, but the responsibility of the maintenance of those policies should help.

Chapter 15

Company relationship with identities

Coming from an identity background, I accept my opinion regarding this subject may be biased. End-of-disclaimer.

15.1 Identity driven organisation

So far, it is common practice to hear statements associating identity with security. These statements sound like "Let's put identity at the heart of security" or "Identity is the new perimeter". We have "Identiy driven security".

In some companies, Security discipline has begun shifting from a cost to a business enabler. So should happen with the Identity discipline. In my opinion identity should guide the organisation decisions. It may sound naïve, but the longer I think about the decisions being made in a company, the more I think they relate to people. Of course it is people who have to take action and endure the consequences of the decisions, but the decisions themselves are very much associated to people: the cost of the salaries of the people working for the company, the assets of the people who have trusted the company with their assets, the markets are a reflection of the societies.

From my perspective, all actors are entities and have identities, which can be modelled, profiled, quantified and managed. Those identities are the forces of nature of the universe of the companies.

15.1.1 Servant leadership of identity management

I truly believe identities should lead businesses, and proper identity management is a key factor in business, as identities include employees, customers, partners, competitors, threats and other stakeholders (which shape the context for the decision making).

A good (marvellous) approach to identity management is almost invisible. Things happen the way This leads to one of the common pitfalls in identity strategy definition: it does not provide visibility.

If there is a place for servant leadership, it is in identity management.

Identities (and their relationships) will shape the future of businesses, but the management of those identities will happen underneath the surface. I have never seen an interview to the project manager of an identity initiative explaining how valuable his contribution has been for the company success, no matter how well (or bad) the project went.

15.2 Relationship with the identity community

Identity management is a complex subject and is full of moving targets. As the whole company relates to identities (at least because they use user accounts to access the tools they use on a daily basis), identity initiatives are extremely complex from both the human-political and the technical perspective.

When an identity position is set up, it is extremely difficult to cover. I have been selected for positions which have been open for more than half a year with no suitable candidate (not sure if that means I am very valuable or that the companies were desperate ;)). I would say that in most cases, there is no suitable individual with all the required knowledge, experience and skills. That is the reason because the community is so important.

One of the most difficult things to find out when working in identity is this: "You are not alone". Even when the size of the identity community cannot compare to software development or agile practitioners (or star wars fans), there is a community working hard on helping each other. When asking a question to a fellow identity professional I have never received any kind of RTFM-like answer; most probably because there is no such manual!

When a company wants to be proficient in anything, it has to have talent related to that “anything”. In the case of identities, it should be part of the community.

The identity-related communities seem to me extremely welcoming.

I will differentiate two kinds (even when they are related): standards committees and working groups on one hand, and professional associations on the other one.

15.2.1 Working groups and standardisation committees

There is major ongoing work relating identities. For just a sample of this (BlockChain and SmartContracts, User Managed Access, etc), you can check Kantara’s groups. These work is being made by people who are passionate about identities and their problems, and the outcome of this work is going to shape the way companies relate to identities worldwide and the

When discussing about standards Vs ad hoc, the dimension that is often missed is the one regarding the influence in the standards committees in order to expand the standards to cover the needs of the company.

15.2.2 Professional associations

There are very few professional associations focused on the digital identity topic. I think this makes these associations much more welcoming and much more valuable.

One of these associations is ID Pro. An organisation of recent creation composed by top level professionals from all around the globe (and mostly from USA).

Beyond specific benefits, such as the typical early access to relevant information, and professional opinions about identity situations, belonging to such a community aids into shaping the future of identities.

15.2.3 What does “belonging to the community mean”

Communities are composed of people. This people are passionate about them and dedicate a significant part of their energy and time to those communities.

When I say a company should be part of the community I mean it should (in addition to paying whichever fees that may be involved) encourage members of the organisation to devote part of their working time to the identity community.

It is not enough for a company to hire passionate professionals and expect them to contribute their spare time to influence the community.

15.3 Who should guide the relationship of the company with the identities?

When companies state they focus on people, they do so either regarding their [paying] customers, their employees or the society. The third case is on purpose ambiguous, so I'll leave it apart. Regarding the other two, we already have the tools to manage (in the best possible way) the information about them that we have.

In most cases, when a company [truly] wants to push an initiative or behaviour in the company, they adjust the organisation chart and provide the budget they see fit.

If a company wants to push the identity initiatives, a figure must be appointed. As to the job title in the badge, to whom should she report and what is the size of the team, that is as difficult to say as it would be to set the priorities of that team. It depends on the company.

One of the options could be naming a Chief Identity Officier, but may be the CXF is already too crowded. There is a good discussion about this here.

In my opinion the role described in the former article misses a huge part of the picture. Even when I think it would be useful to have a figure leading internal identity initiatives related to employees, as discussed in the identity classification section, there are much much more than employees when dealing with identities. Customer Identity and Access Management is a discipline in itself, ant that only covers a small portion of the relationship of a company with the customers. If we add identity profiles for customer segments, competence, fraudsters, cyberattackers, regulators and other actors that the company needs to deeply understand, the task is no piece or cake.

The reason for all these things relating (at least partially) to one role is twofold: it would provide coherence to the efforts of profiling identities, applying AI, managing consents, etc; but more importantly, the role would be able to push the coherence of the identities and their identifiers across the whole organisation.

15.3.1 The Identity Team

The identity team is the set of people whose primary concern are identities and their management. Both concerns are much broader than IT system accounts and both concepts encompass much more than "doing the things".

The identity team is accountable for the definition of the global authorisation model, the global identity management model and setting the rules regarding data retention, permission recertification, segregation of duties and any other regulatory considerations. They are also accountable for the abiding of those rules, but they are not necessarily accountable for implementing all the mechanisms that support identities.

The identity team should set up as little mechanisms as possible for the management and as many mechanisms as required for the auditing and validation of the defined rules. This way of working allows the identity team to be a contributor to the company without becoming a bottleneck and a stopper for the rest of the teams.

The focus of this team is not the operation of the IDM and the implementation of workflows or generating reports about identities, but the identity policies that help the company meeting regulations and doing business.

Some authors such as Alberto Partida assimilates the identity team to the blue team (the team that ensures security), but I disagree. I think identities are much more than security, and security is much more than identities.

If there is a colour for the identity team I think Aqua from the (rainbow series)[https://en.wikipedia.org/wiki/Rainbow_Series] is a good option.

15.4 Compliance is not enough

Isaas Asimov via his character Salvor Hardin said "Never let your sense of morals prevent you from doing what is right!". Well, regarding compliance you may have similar disjunctives.

Compliance to regulations is mandatory, and compliance to standards is a good practice. The way compliance is achieved is where the devil resides.

It is possible to achieve compliance in many ways, and some of them are equally valid from a compliance point of view, but very different from a strategic point of view.

When a new system is added into the card payment process, it has to be PCI compliant. That is good. What could not be so good is to decide that the system should not be created because going through PCI is "too much work". In identities, similar situations arise.

In my opinion, compliance should not be the objective, but the bare minimum, and should only be taken into account once the strategy is clear. Adding the compliance requirements when the strategy is not clear will most likely make compliance the only thought strategy.

In some cases (I am thinking about PSD2 and GDPR) the regulation helps setting a direction of the efforts, but it should not be the target, but the beginning of the road to take. Self-sovereign identity is something to aim for. Our customers will be happier, make better decisions and (hopefully) remain within our company as they feel the respect for their privacy and decisions. In this case I am not restricting the concept of Self-sovereign identity to the common semantic; I am expanding it to the full self-awareness of what our digital identity is and gaining full control over it.

15.5 Standard based Vs Ad hoc is the new Build or Buy

Whenever a new initiative arises there is always the (healthy) discussion about whether the company should build the solution or buy a COTS (Commercial Off-The-Shelf) solution (and most probably heavily customise).

I cannot discuss pros and cons of each approach in this document; I'll just focus on saying that regarding identities, when this discussion arises, it shows an identity strategy that could have been done better.

COTS products provide implementations of the standards, easier integration with other products of the same vendor, plus custom functionalities that have been requested by previous customers or envisioned by the vendor themselves. Ad hoc developments provide all needed flexibility to address current needs.

A truly strategic vision of identities should be such that the standards that COTS solutions implement cover the needs of the company. That way, implementing or buying is not a matter of adaptation of current needs, but a matter of time and cost, plus the ability to being prepared for future challenges.

Having such a vision is certainly difficult, but not having it costs literally millions of euros in personalisation of COTS, building of ad hoc pieces … and doing it all over again at every disruptive change in IT (every 5 years or so).

Chapter 16

Business models around the identity

There are many business models around identities and I don't feel like drawing Business Model Canvasses. I've done it in the past and I can tell it is fun, but for this document I will just focus the discussion on the differential value of the banking sector regarding identities.

I think the differential value that banking can provide can be summarised as:

- Thorough enrolment: banks know their customers very well (at least as much as regulations require)
- Fine tuned processes for defaulter/fraudster/thief identification and management: this is something banks have been doing for a really long time

Of course there are also many other identity-related assets that can provide big value, but if I had to choose two assets I would select the previous two. With other assets I am referring for example to financial information, market information, proximity to the customers, large customer bases, etc.

Once we know what is the value, we have to fill the rest of the canvas. This is something I'll leave to the imagination of the reader with just a few suggestions about options when a customer wants to pay for something in a "shop":

- The customer pays for the use of the banking identifier
- The shop pays for enabling the users to use their banking identities instead of social network identities

- The shop may trust higher transactions made by personally identified users
- The bank could back the identity of the users and claim it is not a fraudster
- The bank could add customer information (e.g. credit score) for the shop to decide payment options (which could also be contracted with the bank itself)

Of course there are the “typical” models around the information enrichment, selling, reporting and the like, but those are not so much levered on the differential value of a banking institution.

Chapter 17

… and so much more …

There are too many identity themes to expand, and I have not yet had time to discuss them all. Here is a little list of what could be expanded (you can think about it as an assorted "to-do list" of pending sections for this document):

- Iterating and optimising identity information
  - Iterating on non-authorisation information
  - Identity governance and optimisation
  - Recertification
  - Iterating analysis of the identity because of
    * New information received
    * Information deleted (right to be forgotten)
- The importance of storing the reason for a change. This applies to the iteration of information enrichment (required for right to be forgotten), manual role assignment (who did it is in audit logs, but the reason may be forgotten at some point), automatic role assignment (a user may have a certain role for two different reasons: because a former boss requested it, and because current department provides it).
- Definition of the identity governance model
  - Iterative and incremental (versus the typical POV stated as "I can only get budget once")
  - Applies to companies where it has to be done "from scratch" as well as companies with "well stablished" processes and systems, as well

as those that say "we have something" and "we know it is much too complex".

- SoD (Segregation of Duties) analysis based on entitlements and systems (not roles). Possibly relate this kind of relationship to the discussions about dependences in the book "Clean architecture".
- Role mining and the relationship between accepting a new entitlement combination with no semantic meaning, and the use of PCA (Principal Component Analysis) in machine learning.
- Auditing and compliance proving
- Discuss about PAM (Privileged Access Management) and how it is just an extension of the rest of the identity management schema, the authorisation model and the only special thing it has regards to access management and auditing.
- AI, machine learning, big data and numeric analysis as tools for data exploring and enrichment / iterating
- Possible use of the identity information for UX: personalisation of the user interface (e.g. show the same controls in the mobile version as in the desktop version for users who "come from" the desktop)
- Solutions for the problems of today Vs solutions for the problems of tomorrow.
  - Deep dive in the false sense of comfort when we design a solution for all current problems, backed up by the arrogance of thinking we already know the full set of challenges the future will bring.
  - Analyse the use of standard solutions as a common solution for the problems and go beyond, making a stand for complex solutions that enable adaptability in the future (e.g. XACML Vs hardcoded policies). Base it in the concept of intelligence "maintaining options open for the future"
- Concept "you build it, you run it" and the impact of the DevOps culture in the classical IAM culture: project decomposition, implementation, iteration, etc.
- Explain the idea beyond Self-sovereign identity: returning the ownership of the identities to the people involved, how privacy should not be an historical anomaly.
- Identity relationships will have an exponential attention increase, and that's good.