Identity management within an organization

Master Thesis Computer Science (Management & Technology)
Radboud University Nijmegen
July 2008

Name: Johan Janssen
Thesis number: 586

Supervisors:

- Erik Poll (Computer Science)
- Ben Dankbaar (Management & Technology)
- Stefan Dijkshoorn (Sponsor from Info Support)
- Marco Pil (Technical supervisor from Info Support)

Preface

This is my master thesis for the Computer Science study at the Radboud University Nijmegen. This research combines both security and management as I followed the Management and Technology master track. The research was conducted at Info Support in the Netherlands.

First of all I would like to thank all the people at Info Support including the other students. The people at Info Support offered excellent guidance during the project which made things easier for me. The last half year was a pleasant and interesting time where I learnt a lot.

Then I would like to thank the supervisors from the University. Erik Poll was involved from the security department and Ben Dankbaar was involved from the management department. The regular discussions with them made sure that I knew which direction to take and what to do to complete this thesis.

Last of all I would like to thank my family, friends and girlfriend for their support and patience during my six years at the University.

Abstract

Organizations have grown over time and so has the number of software applications they use. Not only the number of applications but also the number of users that need access has grown. Suppliers and other partners also want to access resources from within the organization. For a couple of years there were no strict access rules, the people who had access to a computer could access all resources. Over the years the number of applications grew and companies started to realize that they had to protect their resources. That resulted in applications with their own authentication mechanism; an employee needed a username and password (identity) for every application. With the growing number of applications the number of usernames and passwords an employee had to remember also grew. The result was that the maintenance of all those identities became more complex. Users needed to remember all the identities. Administrators had to maintain all identities and the access rights belonging to those identities. Management could not really understand those access rights so they were unable to verify things such as privacy protection and they could not hold employees accountable for their tasks when the employees did something they were not allowed to.

Identity management can help to solve the problem above. The idea behind identity management is to centralize identity and access management. Instead of many applications with their own authentication and authorization mechanism identity management is centralized. The centralization can be constructed with a LDAP server which is a central place where the usernames and passwords are stored. That server can be used to authenticate and to define the access control.

The thesis consists of two parts a managerial part and a technical part. These are combined into one thesis but are mainly treated in separate chapters. In the thesis I have tried to find an answer to the following two managerial questions:

- What are the benefits for organizations when using identity management? Or in other words why should an organization opt for identity management?
- What are the considerations for organizations when using identity management? Or in other words, what should the organization do when introducing identity management?

At this point it seems that the problems that companies have with identities and access control can be easily solved with identity management. There are however two problems: companies do not realize the benefits of identity management and/or they implement identity management in a 'bad' way. The problem is that most companies cannot see direct value of identity management, the costs are spread across the company and it is hard to make them explicit. Reduction of costs should not be the (only) driver of identity management. There are more benefits such as improved security, user convenience and the ability to allow other organizations such as suppliers' access to specific resources of the company. However these benefits are unclear for many organizations and they do not implement identity management, or they implement it because it is required by law or legislation. When the management does not understand the clear benefits of identity management then the support from the top level of the company will be low. That will result in employees who will not be too enthusiastic. In the end that could result in identity management that is not well implemented and cannot realize all the benefits. As identity management becomes more and more important and organizations start to realize that it is not only a technical thing, it was interesting to see what the current developments are.

It seems that the organizations start to realize that identity management should involve management, administrators and users. They should work together to define

policies, processes and the technical implementation. There is no straightforward solution to introducing identity management. As identity management involves many aspects and is closely related to the organization's structure (for the access rights) and the organizations applications (for the authentications) it is very organization specific. But there are some guidelines and best practices that can be used to introduce identity management.

This thesis consists of two chapters that are mainly managerially orientated namely chapter 3: 'Business drivers for identity management'. This chapter explains the main drivers for an organization to spend time on identity management. There are quite some advantages of using identity management which are discussed in this chapter. Then chapter 4 'Identity management in a business environment' shows how it comes that some companies end up with 'bad' identity management. To try and give some guidance to companies to avoid 'bad' identity management the rest of the chapter is dedicated to treating the issues one should keep in mind when introducing identity management.

After the managerial part comes the technical part where I tried to find an answer to the following question:

- Is .NET or Java better suitable for authentication and authorization with an LDAP server?

Some organizations have a policy which describes the language to use; other organizations do not have a strict policy about the programming language. If there is no strict policy then it might be interesting to see if some language is better suited for identity management then another language. In this thesis the differences between Java and .NET are analyzed. The conclusion is that it is possible to implement identity management in both languages. The languages have some differences such as the available documentation, dependency on operating system and the level of abstraction but in the end they are both quite suitable. When choosing between the languages it is best to look at the expertise within the company and the configuration of the network. If there is more expertise in one language then that should be the language of choice. If you have mainly Microsoft products then .NET is probably the best choice and if that is not the case then Java might be the better choice. The question however is if it is practical to implement identity management from scratch or if it is better to use a standard package. That is because identity management can get quite complex and it has to communicate with all applications that you use within the organization. Building something that big might prove more costly in the end then buying a standard package and customizing it to your needs.

Table of content

1 Introduction

Identity management is a hot topic for lots of organizations, but there are some obstacles to overcome before it can be used effectively. The intention of this thesis is to assist organizations in the implementation process of identity management. It might also be useful for organizations that already have an identity management system but want to improve it.

Maybe you have heard about identity management before, but what is it exactly? To answer that question I have included two definitions:

- "*Identity management is the set of business processes, and a supporting infrastructure for the creation, maintenance, and use of digital identities.*" (1)
- *"Identity and access management refers to the processes, technologies and policies for managing digital identities and controlling how identities can be used to access resources."* (2)

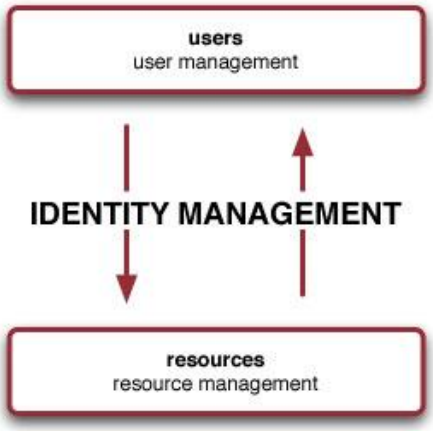


Figure 1 Identity management: manage users and resources[1]

As these definitions show it is not only a technical problem it is also an organizational problem. Business processes, policies and technology should be aligned to maximize business benefits. Security is one advantage of using identity management but there are more benefits for organizations.

Security was already a topic for 'Bestuurlijke Informatiekunde' (3) but identity management became only popular in the last few years. Information systems are more and more digital and very critical for many companies. It is necessary that they keep working under all circumstances. Another common reason for using identity management is compliance to rules and legislation. That is because law and legislation require transparent processes to ensure privacy and accountability (4).

There are four different types of security measures that a company can take:

- Preventive (to prevent problems from happening)
- Signaling (to signal security breaches with for instance logs)
- Repressive (to restrict damage as much as possible)
- Corrective (to restore the damage that is done)

Identity management is mostly seen as a preventive measure. You deny users from accessing resources where they have no authorization for.

[1] Source: http://www.direxon.com/index.php?id=36&L=2

Nowadays identity management is a hot topic but in 2003 only 25% of the businesses where planning an integrated secure identity management solution in the near future (5). With an integrated secure identity management they mean a complete identity management solution that works with the existing applications.

In the past identity management was mostly seen as something application specific. Every user had a login name and password for every application he or she used. Today there are so many users, not only employees, but also external parties such as costumers that need access to information resources from the company. The number of applications within the organization is also quite large making it harder to maintain the access rights the users have to all those applications. That together with laws and regulations had a big impact on identity management. It is no longer just a technical problem it involves the business processes as well.

In Figure 2 below you can see the users, both internal (top left) and external (top right) and the resources they want to access. This gives a good overview of what identity and access management is about. Each identity has specific access rules and depending on those rules each identity can access several resources such as applications and services. This shows that it is currently a lot more complex than just denying users access to some directories or applications, which was done in the past.

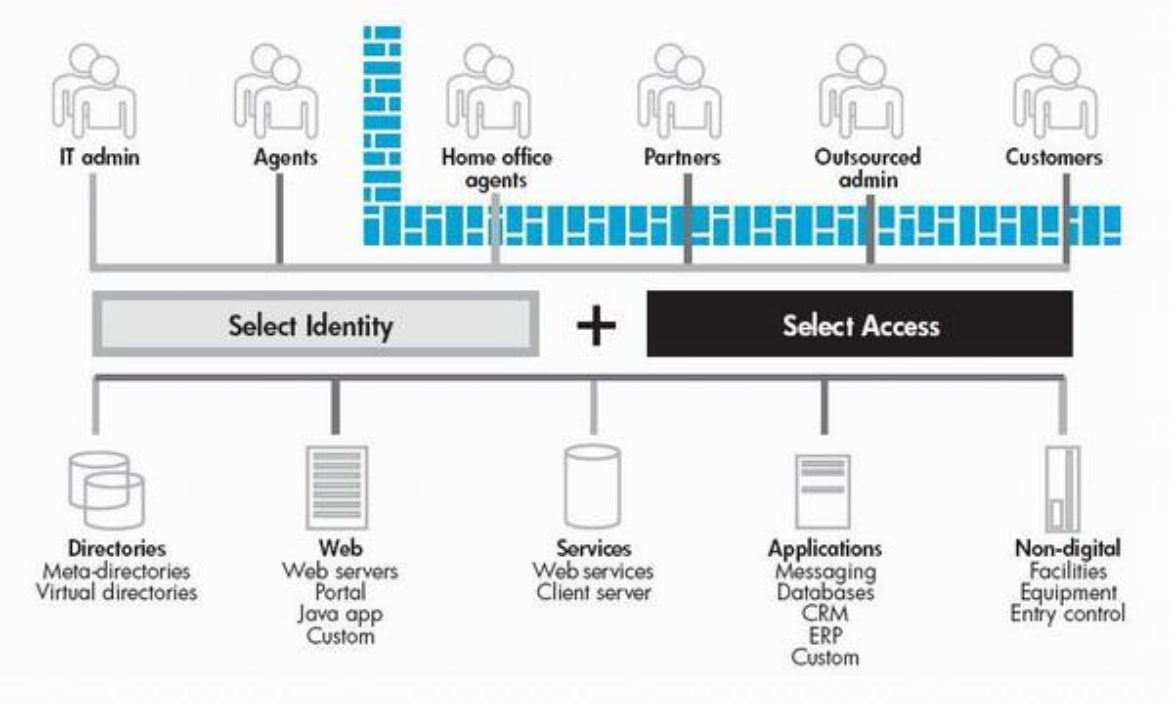


Figure 2 Identity management organization[2]

Current research for example the Quest survey from 2008 (6) shows that IT professionals still regard identity management as a hot topic. Figure 3 and Figure 4 show that a large percentage of the IT professionals think that identity management is important within their organization. The Quest survey also shows that 71,7% of the IT professionals believe that identity management will become more important within their organization in the next five years. A report (7) by Global Industry Analysts

[2] Source: http://www.nsai.net/services/identity-management.shtml

estimates that the identity management software market will reach $4.9 billion by 2012.

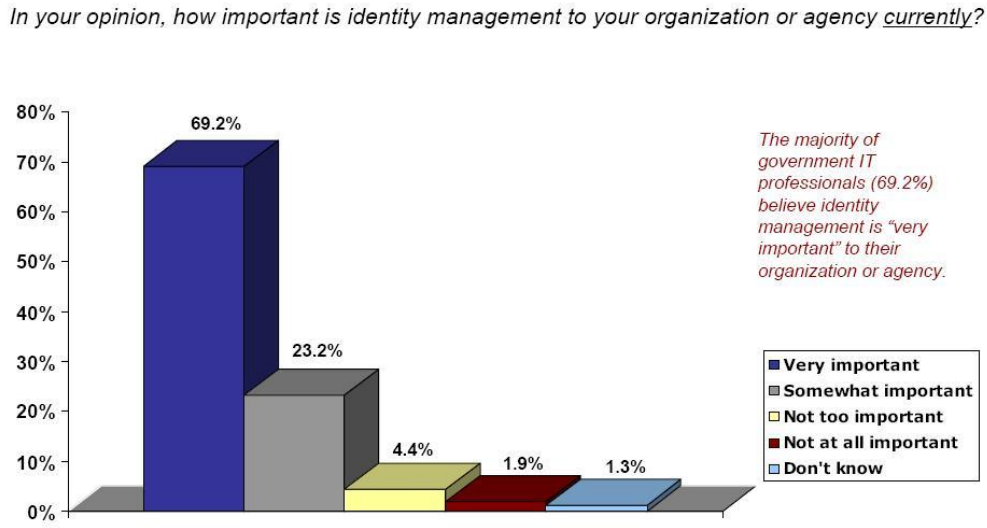


Figure 3 Importance of identity management(6)

The problem however is that a lot of organizations still do not have an identity management system. Figure 4 shows that only 19,1% have an identity management system and 55,9% are instituting one. It is important to implement identity management in a correct way to benefit as much as possible. The law can even require identity management systems and for the organizations there are other business drivers to consider before implementing identity management. That compliance is an issue can be seen in the survey; 37,1% is not sure when they are compliant with federal mandates and only 14,8% is compliant already. Organizations are required to become compliant, so some work needs to be done. To make sure that the process is executed well I hope to provide some assistance with this thesis so companies are not only compliant but also get the other benefits associated with identity management.

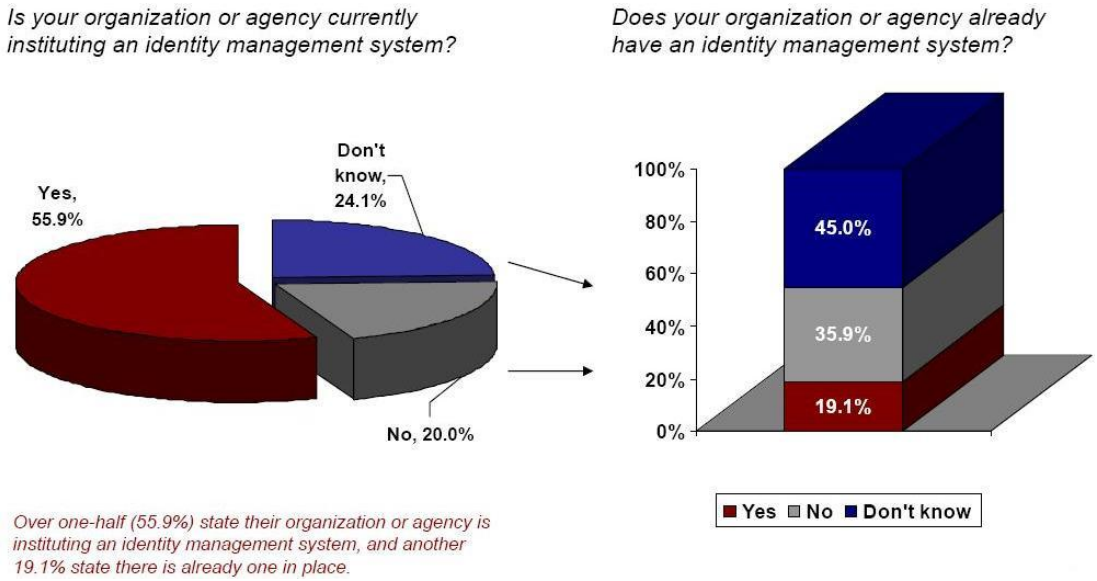


Figure 4 Usage of identity management(6)

In this thesis the term identity management will be used quite often. The term identity management is often used for the term identity and access management in the literature. So identity management in the literature does not only cover the topic of identities but also the access controls for those identities. In this thesis I will use the same convention and if I mean 'pure' identity management in this thesis then I will mention it explicitly.

1.1 Problem description

Identity management has grown over the years. Organizations have some ad-hoc implementation which makes it hard to maintain. If you do not use proper identity management then every system will possibly have separate authorization tools, administrators and business process. Every application has different IT processes and business processes to maintain the identity management for that application. That makes both the technical and the business part very complex which results in more difficult maintenance then necessary as illustrated in Figure 5. There are lots of links between different parts of identity management. Especially the link between business processes and IT processes is made with numerous links. For example a user has a list of various authorizations and when those authorizations change then a lot of those links have to change. So when a user switches function within the organization then you cannot just change the function in the IT system, you have to set all authorizations separately.

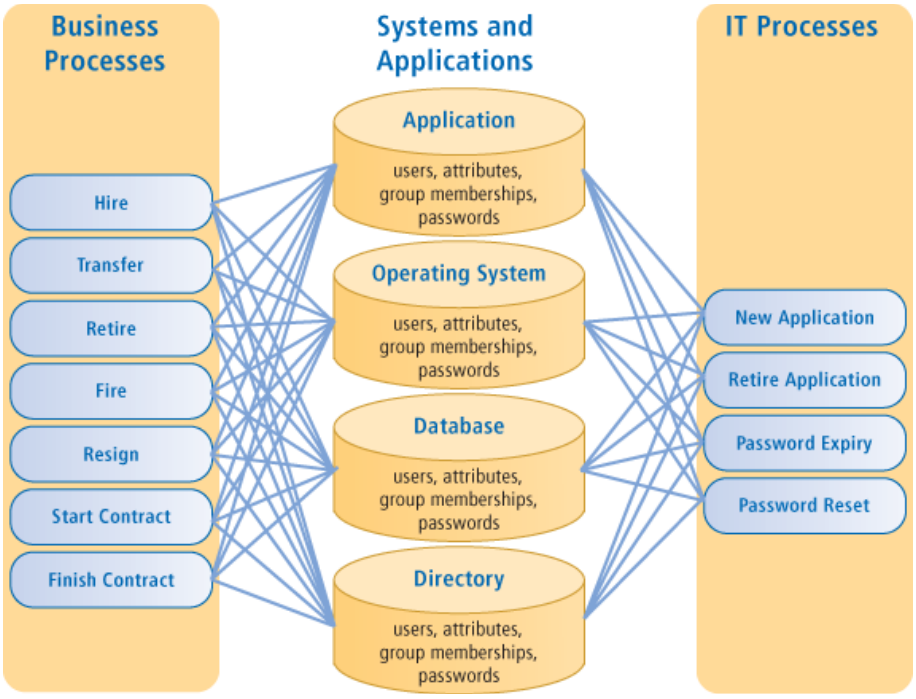


Figure 5 Business without identity management (8)

When using identity management the existing authorization tools and different processes within the organization are replaced by one uniform tool and uniform processes that are used within the entire organization. Those are maintained by one (department) of administrators. That makes the systems and processes more transparent which facilitates the business, supports system administration and offers

the users more support. The usage of identity management is shown in the following picture. With this new model you can see that there are less links, which makes maintenance a lot easier. The example mentioned before about a user switching function now only requires to change the function of the user in the IT system.

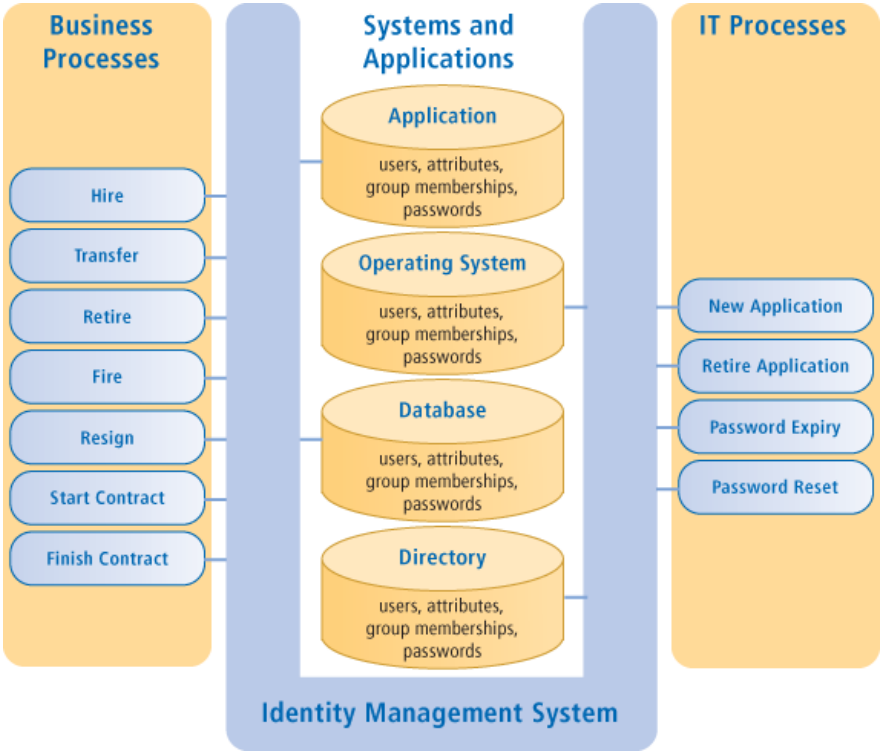


Figure 6 Business with identity management (8)

The solution to the so called spaghetti of authorizations can be solved with identity management. But over the years it became clear that identity management has its own problems. Various surveys indicate those problems, for instance a survey from KPMG (9) shows that:

- A lot of organizations do not have a grip on identity management.
- The media tell us that cost reduction is the main driver for identity management but that is not true. User convenience and compliance with legislation and regulations is valued much higher.

These problems indicate that identity management is not introduced properly and that the benefits are still not clear. As shown before identity management is or will be introduced in many companies. Those companies will benefit from a solid introduction, but they will need some guidance. Administrators often do not have expertise in identity management and receive minimal support from management since identity management is seen as something technical.

Companies might need identity management because of laws and legislation, or they want to introduce it themselves to realize the business benefits. More information about the different business drivers can be found in the chapter dedicated to that subject. However identity management introduces some problems. First of all the organization has to realize that identity management is an organizational problem and not just a technical problem. Management should set up identity management and that should be supported with a technical implementation. It is important that the process of introducing and maintaining identity management within the business is

executed carefully. An ad-hoc implementation will be harder to maintain and extend, a solid base will provide easier maintenance and make it easier to extend. The technical implementation should be done in steps and based on standards, frameworks and guidelines to ensure interoperability. Especially in this time where companies work closely together and takeovers happen regularly it is important that systems can be coupled together.

Then there are the companies who do not want to introduce identity management. They think that identity management is not important enough to spend time and resources on. That is not strange since it is difficult to make the benefits explicit. The costs for identity management are spread through different divisions. The systems also work without identity management so there is no need to change. Responsibility is also an issue, since identity management is spread across divisions it is not clear who is responsible.

These problems can be grouped together in the following list:

- The business benefits of identity management are difficult to explain.
- Management thinks that identity management is just something technical.
- Administrators are no identity management experts and find it hard to implement it without guidance. This eventually can result in systems that are hard to maintain.

1.2 Research goals

The goal of this thesis is to assist organizations with their identity management. The thesis should provide organizations with possible reasons to implement or improve identity management. Further the thesis should help organizations with the actual implementation. This will be done from a business perspective and a technical perspective. The technical part of the thesis will consist of a comparison between java en .NET for identity management with LDAP. LDAP is some kind of telephone book with usernames, passwords and more information. LDAP can be used to maintain the users within a network and give them certain permissions. Does Java or .NET provide certain advantages regarding functionality, usability etcetera? With that comparison organizations can decide whether they would like a Java or a .NET implementation of identity management.

The goals can be described as follows:

- Provide a list of business benefits when using identity management.
- Provide assistance to companies when introducing identity management.
    - Provide guidance for setting up their business processes.
    - Provide a comparison of Java and .NET for the technical implementation.

1.3 Research questions

In the previous paragraphs I explained what the problems are with identity management. To help organizations with these problems I want to provide organizations with the information that is necessary to set up identity management in a successful way. In this thesis both the managerial and the technical part will be treated. There is not much information about the specific advantages of Java and .NET with regard to identity management through LDAP. Organizations might be interested in the advantages or disadvantages of each platform when they decide which one to use for identity management. That is why I will describe the characteristics of each platform.

The research concentrates around the following three main questions:

- What are the benefits for organizations when using identity management? Or in other words why should an organization opt for identity management?
- What are the considerations for organizations when using identity management? Or in other words, what should the organization do when introducing identity management?
- Is .NET or Java better suitable for authentication and authorization with an LDAP server?

1.4 Structure of this thesis

In this paragraph the structure of the thesis is explained. The thesis consists of two parts, the business part and the technical part. It is possible to just read the part where you are interested in. The business part is discussed in chapter 3 and 4 while the technical part is discussed in chapter 4 and further. Chapter 2 contains the context and explains some topics that are vital for the further understanding of the paper.

Chapter 2 contains the context of identity management. Some important concepts are explained and it should serve as an introduction for the rest of the thesis. If you are not familiar with these concepts then you should read this before the business or technical part as those parts assume that you know the concepts from this chapter.

Chapter 3 treats the business drivers for identity management. Some reasons are given why an organization should introduce identity management. It is important that the organizations see the benefits of identity management so that the implementation is supported by management and executed in the best possible way.

Chapter 4 explains the business environment or what an organization has to know before and when implementing identity management. Introducing identity management is not something simple, both the business and the technical people have to work together to realize all the benefits of identity management. In the past quite some projects where executed and that resulted in useful information that can be used so that one can benefit from the results of the past.

Chapter 5 treats directory servers which are used for the Java and .NET comparison. I had a look at some directory servers and made a comparison to see the differences.

Chapter 6 shows how Java authentication and authorization service works. This is the standard to use in Java if you want to use authentication and authorization. The examples use a directory server for authentication.

Chapter 7 shows how authentication and authorization in Microsoft .NET works. This is done to compare Java with .NET and see what the differences are. That might have an influence for companies that still have to choose a platform.

Chapter 8 is about Service Oriented Architecture which nowadays is often used when creating applications. Previous chapter treated some quite simple identity management implementations. This chapter looks at some more advanced identity management with a Service Oriented Architecture that is used a lot these days. Again this is done to compare Java and .NET.

Chapter 9 treats federated identities, those identities can be used outside the companies walls. As employees now work from their homes, suppliers want access and so on, identity management has to change. Federated identities offer another challenge for identity management; it should offer the organization greater flexibility

but it is not trivial to implement. This chapter will also be used to show the differences between Java and .NET.

Chapter 10 discusses the differences between Java and .NET regarding authentication and authorization with an LDAP server. One would expect that they are quite similar, but do they really offer the same quality characteristics like functionality and usability?

Chapter 11 is about guidelines for the technical implementation of identity management. As identity management is difficult to introduce it would be nice to have some guidelines about what you should do and what you should not do. This chapter gives an introduction to these guidelines that can be used to assist in the implementation of identity management.

Chapter 12 contains the conclusion of this thesis and recommendations for further research.

2 Context of identity management

In this chapter the key technical concepts and some non-technical concepts of identity management are explained. In the following figure you can see identity management and some of the technical concepts related to it. These are the essential concepts behind identity management. Some of these concepts are explained in this chapter others will be discussed in the rest of this thesis.

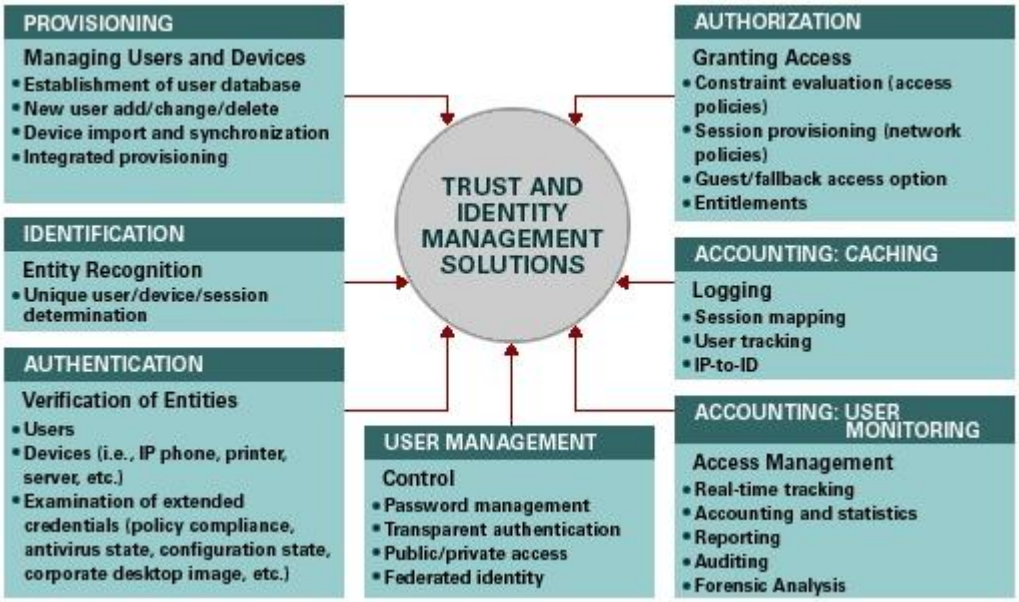


Figure 7 Identity management context[3]

2.1 Digital identity

Everybody has some sort of identification like a passport, driver license etcetera. An identity uniquely defines the identity of an object within a context. In a company every employee has a unique personnel number. That number is unique for one person and can be used for identification.

Another 'definition' can be found on the Oracle(10) site where identity is described as the relation between the user and:

- Relationships
- Resources (credentials, privileges, activity)
- Authorizations
- Personal & Corporate Information
- Roles

Digital identity is some digital information about a particular person. In order for digital information to be regarded as a digital identity, the data should be usable to determine who you are.

Digital identity can be seen as a composition of the following parts (11):

- Identifier (uniquely describes the person, such as e-mail address)

[3] Source: http://www.cisco.com/en/US/netsol/ns463/networking_solutions_sub_solution_home.html

- Credentials (can be used to prove the authenticity of the person, such as a password)
- Core attributes (can be used in multiple contexts, such as the address)
- Context-specific attributes (can be used in a specific context, such as a workroom number which is only relevant for employees)

A digital identity can provide data that shows that you belong to some kind of group. Like a system administrator and that you have the rights belonging to the group you are in. So it is not only valuable as identification, we can also use it to give people access according to the group they are in.

In practice a digital identity is mainly composed of a username and password and all other personal information that a company requests. That personal information can include the date of birth and information such as where the person lives.

2.2 Identification

Identification is the process of identifying the identity of a user who tries to access a system. People can identify with numerous things, a passport, identification card, college card or drivers license are some common identification methods. The data on those identifications are compared against yourself.

2.3 Authentication

Authentication is the process of proving your identity or in other words to prove who you are. It is a method to establish that someone is the one he or she claims to be.

That can be done in three different ways:

- By something you know: such as a password.
- By something you have: such as an access card.
- By something you are: such as a finger print.

Authentication is often used to access buildings, e-mail and other protected resources.

2.4 Authorization

Authorization is the process of giving access to specific resources to people or systems. At the authorization stage the system gives the user access to the resources the user is entitled to.

Authorization is mostly preceded by authentication; after a user is authenticated the authorization step checks what the rights are for the user are. Authorization can then give access to applications, data and physical components such as printer's etcetera.

Authorization can be done separately, for instance by giving every person in the organization separate rights for every application. A more efficient way is group based, for instance administrators and users have a different set of rights. Or you can give rights based on the task people have to fulfill within the organization, such as checking the administration.

Working with groups or tasks is also known as role based access control (RBAC) where users are divided in groups according to their role within the organization. In that case you can give everyone in the same group the same rights, which are easier to manage then separate rights for everybody.

2.5 Access control

The aim of access control is to make sure that users only have those rights to access resources that they need to fulfill their function (12). Access control takes care of confidentiality and integrity for the system. With access control users can only read or write to resources if they are authorized.

Access control is more or less the implementation of the security policy. In that policy is described which rights an authenticated person has (or does not have). So access control can be used to figure out what the rights of the corresponding user are.

Mostly the above concepts are used together, although there are some variations. The following stages commonly take place when accessing a resource:

- User tries to authenticate.
- User tries to authorize: authorization communicates with access control.
- User gets access or is denied access to resources.

Another useful option is logging all access requests. Then afterwards someone can be held accountable for his actions.

2.6 Provisioning

Provisioning is the creation, maintenance and destruction of an identity and setting the attributes belonging to that identity. Self-service provisioning is a provisioning system where there is no action necessary from administrators or the helpdesk. People can just do it themselves; an example is the creation of web based e-mail accounts at Gmail and Hotmail.

2.7 Information policy

Information policy is the adjustment of information provisioning to the information technology and the company policy. In other words it is a way to figure out how the information technology can help support the company policy with the help of information provisioning in the best possible way. Information policy is a concept of 'Bestuurlijke Informatiekunde' (3). For the information policy it is necessary to describe the quality one wants together with the resources one needs.

The information policy should answer the following questions (13):

- How to deal with company data?
- How to deal with information systems?
- How to deal with information technology?
- How to deal with the working environment surrounding the information provisioning?
- How to deal with resources necessary for the information provisioning?

The information policy will give us the following answers (13):

- Targets and choices about how to handle the above points.
- The conditions set for the realization of the information policy.

Policies in identity management are more focused towards how to deal with access to digital resources. They can for instance describe the access rights belonging to a particular function.

2.8 Identity management

Identity management is an abbreviation for identity and access management. In the literature you mainly see identity management, but what they (mostly) mean is identity and access management. It is the combination of processes, policy and technology for the use and management of digital identities. Where identity management stands for the management of digital identities and access management uses the digital identities for access control. In this thesis I will also use the term identity management instead of identity and access management.

In previous pictures identity management was described quite abstract or by concepts. The following gives an overview of how all elements relate to each other. First you have the external and internal users and application. They communicate with the access management and identity management who in turn communicate with directories and identity provisioning. Those identity management services are monitored and audited. At the background there is the communication with resources like applications and systems. Some terms like SOA and RBAC are described later in this thesis.

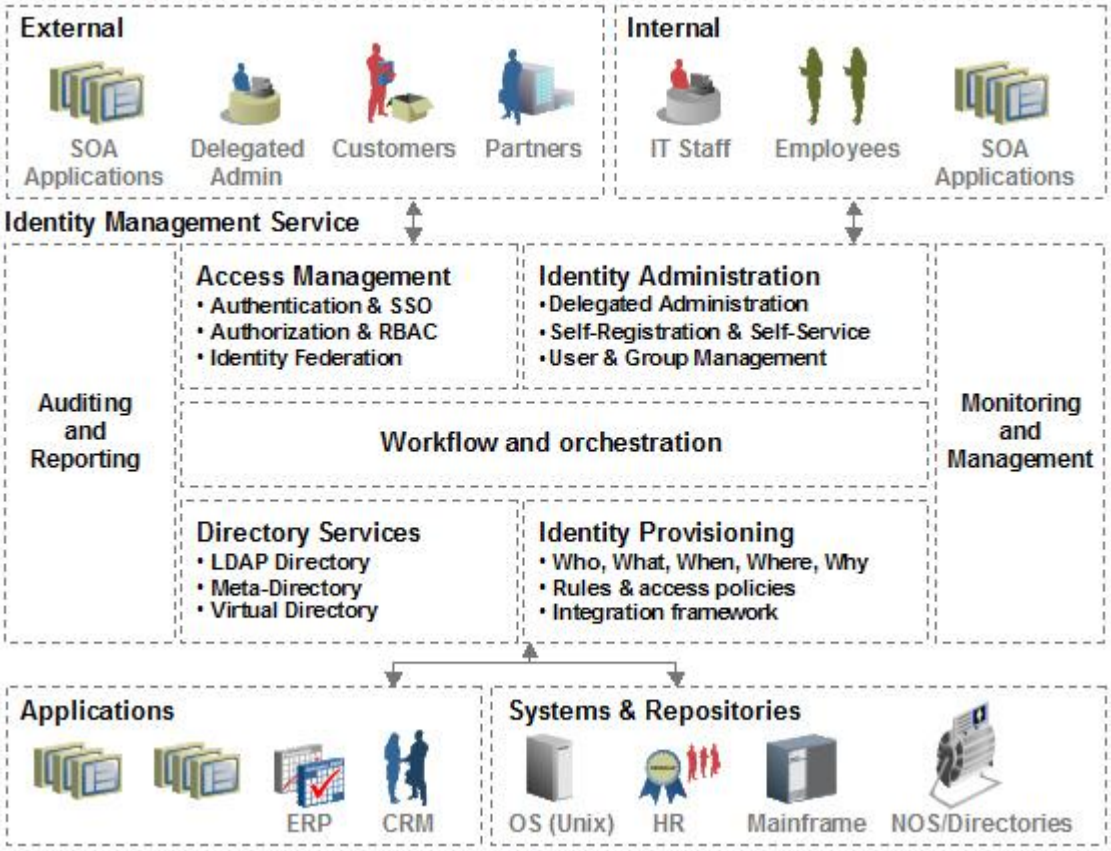


Figure 8 Identity management overview[4]

There are multiple descriptions and definitions of identity management. One of them is the following: Identity management contains the processes and all underlying technology to create, maintain and use digital identity data (14).

[4] Source: http://blogs.oracle.com/schan/newsItems/departments/userIdentityManagement

Although there are numerous explanations of the term identity management most rely on the essentials elements of an identity management system:

- Policy (for instance: who has access to which information).
- Technology (systems and applications).
- Process (for instance: how do you request authorization).

Those three elements are necessary to set up a good identity management system.

Policies control the access rights; the policy contains the rights that the users of the system have. There are many different rules, for instance access rules to network shares or rights to access company information from outside the company.

The organization should define processes that should contain guidelines about what to do in different circumstances, or what to do when new resources or users are added to the system. Another process description should be given about what to do when someone requests authorization.

The technology part consists of three main elements:

- Identity life cycle (the cycle an identity goes through from creation until destruction).
- Access management (the system to give users authorization to the resources they are entitled to).
- Directory services (the system where users and their credentials are stored, it is used to authenticate but can provide a number of other services).

Managing the identity life cycle is necessary, because the identities and roles of persons within an organization change over time. People can get another role or even leave the organization. For all those actions processes should be defined and a technical implementation should be available to realize those processes. An example of those processes is the process when someone requests (more) authorization.

Figure 9 Identity life cycle

The identity life cycle consists of several stages, varying according to their detail. Above is an example that shows the identity lifecycle from the initial hire of employees until the termination of access rights when the employee leaves the organization.

2.9 Federative identity

Years ago people were happy if they could just work within their company. Nowadays users want to work at home or access resources through a third party. That third party does not know which access rules you have and maybe you do not even want that third party to know your identity.

Federative identity is a solution for those problems. With federative identity you can try to login to a system that does not know your identity, the system can then ask a system that knows your identity for information and give you access or not. The third party does not even have to know your identity; it can trust on the system that knows your identity and use the rights that it gets from that system. An example is the GSM network; you can connect to it using the network from your provider, but also networks from other providers. Those other providers check with your provider if you are allowed to use the network and they bill your provider if you use it.

2.10 Identity 2.0

Identity 2.0 refers to Web 2.0. It is a new way of managing identities. In the past identities were mostly created for every single resource and maintained by the organizations that issued them. That is not really user friendly; people have to remember many usernames and passwords. With identity 2.0 the user is the central point instead of some resource. The user is now responsible for his identities instead of all the organizations. In that way users can use one identity to access multiple resources. That is possible with programs like openID and Windows CardSpace. Those programs store the identities that you have and you can access them with one single identity. When trying to access a resource you only need the identity to access openID or CardSpace once and then those programs handle the authentication for the resources.

2.11 Single sign-on

Single sign-on (SSO) is a form of access control, with SSO users can authenticate once and get access to multiple systems. It is closely related to Identity 2.0, SSO can be seen as an implementation to realize the ideas behind Identity 2.0. SSO has many applications, it is not only used within companies to allow users to login once and then use all applications that they need without having to authenticate for every one of them separately. People can also use it to access web sites, applications and other systems. With programs like OpenID and Windows Cardspace you can store multiple identities for the various applications you have. In those programs your identity including your password is stored. When you start your computer you authenticate to one of those programs and the program then authenticates you if you try to access a resource.

SSO is a popular topic in identity management, as the survey by Quest (6) shows that 59,3% of the organizations use SSO in their identity management strategy.

2.12 Quality aspects

After this chapter some comparisons are made between Java and .NET for instance. The target is that we can identify different characteristics of the two. These characteristics can have an influence on the choice of language for identity

management development. These characteristics show the quality of the software, maybe Java has more functionality and .NET is more user friendly. What quality is exactly is subject to discussion, it is quite subjective. Quality of software can be described by various characteristics of software. To compare Java and .NET I will evaluate some of the software quality characteristics that I found in the literature.

There are many different concepts about software characteristics in the literature such as the tree of Boehm which is a concept of 'Bestuurlijke Informatiekunde' (3). The tree consists of the following components and sub components:

- Usability
  - Reliability
  - Efficiency
  - User friendliness
- Maintainability
  - Testability
  - Transparency
  - Changeability
- Transferability
  - Completeness
  - Independence of equipment
  - Independence of organization

There is also the ISO 9126 standard, which is a standard for the evaluation of software quality[5]. ISO 9126 contains the following characteristics and sub characteristics:

- Functionality
  - Suitability, Accuracy, Interoperability, Compliance, Security
- Reliability
  - Maturity, Recoverability, Fault tolerance
- Usability
  - Learn ability, Understand ability, Operability
- Efficiency
  - Time behavior, Resource behavior
- Maintainability
  - Stability, Analyzability, Changeability, Testability
- Portability
  - Install ability, Replace ability, Adaptability

In the 'kwaliteitsdriehoek' described by Bemelmans (15) some other characteristics are described: flexibility, reusability, correctness, integrity.

In 'Grondslagen administratieve organisatie' (16) and 'Inleiding EDP auditing' (12) some of the quality characteristics I described above are also mentioned.

In the rest of this thesis I will use some of these characteristics and try to compare Java and .NET. That way I try to figure out if one of those platforms has an advantage above the other.

[5] See also ISO 9126 on Wikipedia: http://en.wikipedia.org/wiki/ISO_9126

3 Business drivers for identity management

Why should organizations use identity management? There are numerous reasons and the reason can have an impact on the process and implementation of identity management. In the end a business wants to maximize the business value, identity management can help to realize that. Some of the reasons for identity management are treated in this chapter. Identity management can start quite simple, but can result in an increase in business value as shown in Figure 10. In the picture below you can see that 'standard' identity management provides business value, but with more 'advanced' identity management you can gain even more business value.

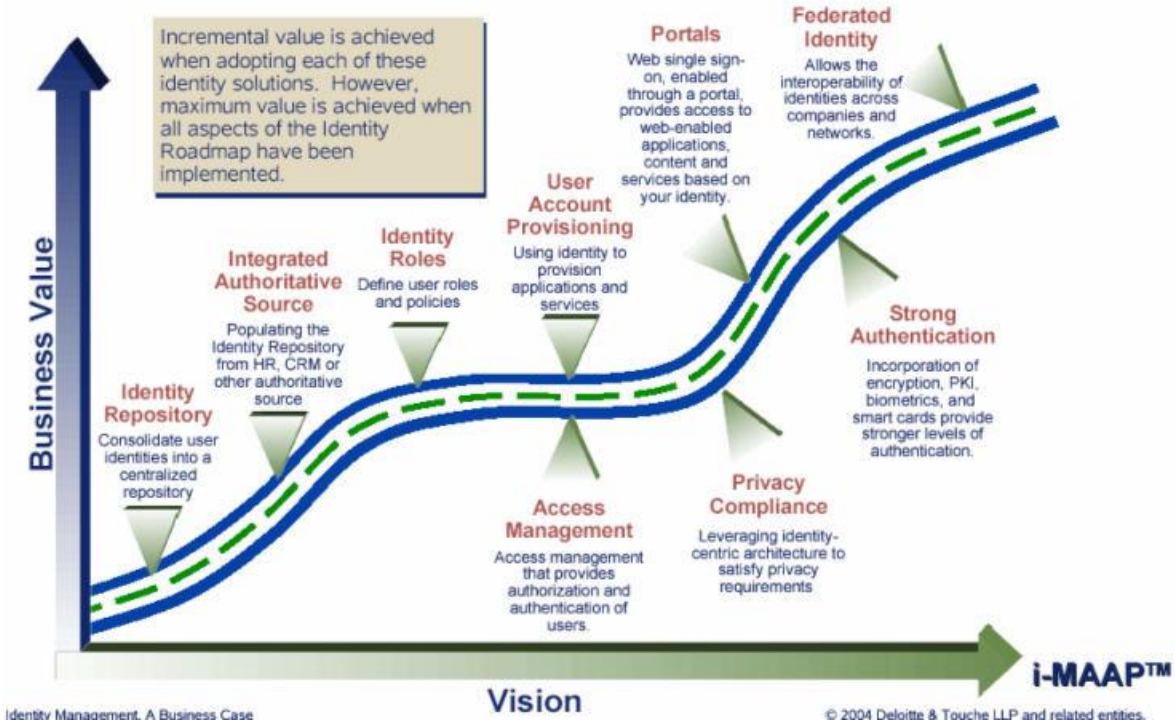


Figure 10 Identity management roadmap[6]

In the last couple of years companies got an ever growing portfolio of applications and systems. Managing who has access to which resources is a difficult task, but why should a company spend time on it?

The reasons for identity management are numerous and can be divided in some partly overlapping categories. To give an introduction I will present a list first and describe the reasons in more detail in the following paragraphs. The list with some of the reasons found in the literature (5), (8), (17):

- Business facilitation
  - Reach global customers
  - Tighter supplier relationships
  - More productive partnerships
  - Ensures a companywide policy
  - More flexible infrastructure
- Increase security
  - Consistent security policy

[6] Source: http://www.provost.utoronto.ca/public/reports/overview/appendices/a.htm

    - Immediate system wide access updates
    - Consistent identity data
- Cost reduction
    - Eliminate redundant administration tasks
    - Reduce help desk burden
- Increase productivity
    - Fast employee ramp-up
    - Free-up admin staff for strategic projects
    - Single sign-on
- Service level
    - Focused personalized content
    - Comprehensive profile view
    - Self-service
- Regulatory Compliance
    - HIPAA/Privacy Act, Graham-Leach-Bliley

There are lots of reasons but there are some reasons that are regarded more important than others. Compliance is something all companies have to do while the service level is something optional. That some reasons are seen as more important within companies is also visible in the surveys. According to a survey from Quest the top reasons for instituting identity management are (6):

Reason	%
Increase physical, data and information security	33,3
Compliance	32,1
Protection of personal information	19,0
Simplify internal systems	2,5
None of these	3,8
Do not know	9,3

The survey from KPMG (9) shows that companies find cost savings not really important. User convenience and compliance are rated as reasons to change. The results of the two surveys show quite some difference as the Quest survey does not really indicate that user convenience is important, although simplify internal systems might be about user convenience. The numbers show that security, compliance and user convenience are very important to companies. Companies might choose to implement identity management only to realize the 'main' benefits, or they can choose to implement it as broad as possible to gain other advantages.

Based on the reason why a company wants to introduce identity management the actual implementation may vary. For some reasons you do not need to implement all identity management aspects. It is good to know what is necessary to reach the goals set for identity management. That way you can introduce the things that you need immediately and later extend the identity management system with other parts.

In the literature (17) I found a pyramid which shows some business drivers like compliance, risk reduction and increased productivity together with the necessary implementations. Figure 11 shows that the various elements of identity management can be build on top of each other to reach a certain level of identity management that is required by the organization. When you want compliance you need audit trails and all underlying concepts, but when you just want fundamental identity management, something like audit trails is not necessary. So if a company wants to realize a certain level of identity management it can see what concepts are necessary to achieve that level.

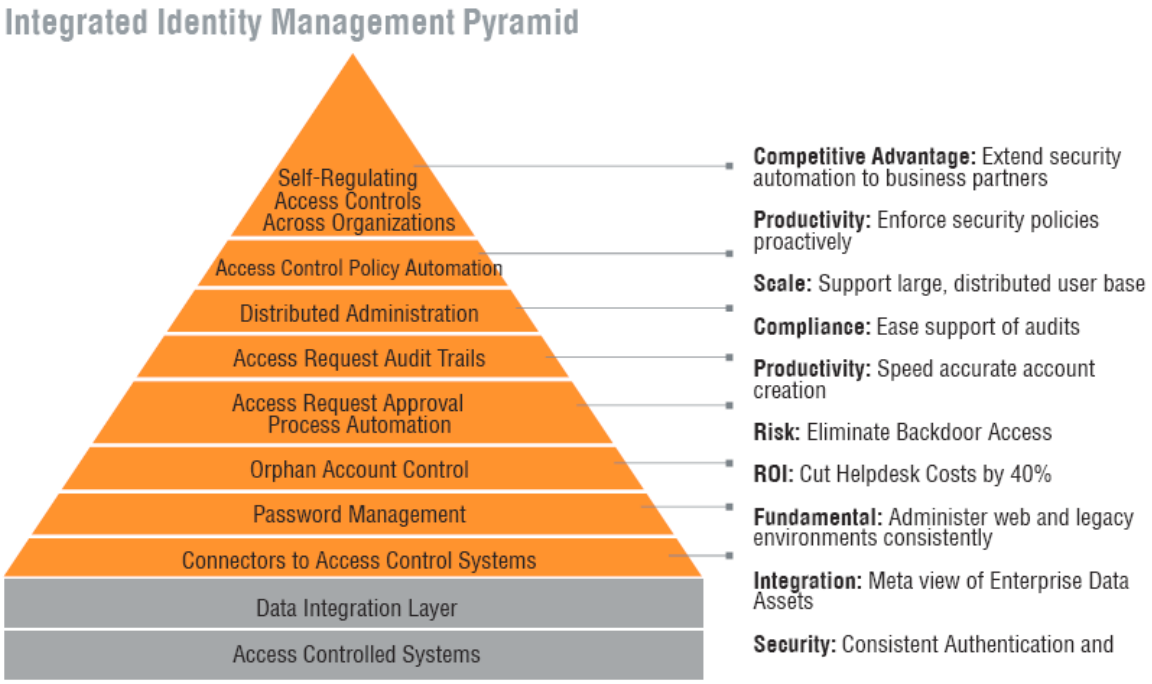


Figure 11 Identity management pyramid (18)

3.1 Security

As shown above the surveys show that security itself is one of the most important reasons for companies to use identity management. That has various reasons, for instance to protect against so called hackers. But for companies the existence of hackers is not their only concern. Their own employees are one of the biggest threats. They can access sensitive information and distribute it. Or use their computer facilities for so called unauthorized use, to use company resources for private purposes for instance.

Another problem is that users often share their credentials (password etcetera) because authorization requests take too long or there are many different passwords for all systems. The survey from KPMG (9) shows that there are some various security threats within companies. One of them is that users share credentials, the survey shows that only 21% never share their credentials. Another issue exists when users have too liberal authorization according to 37% the authorizations are too broad and only 27% think they are not too broad. For compliance and maintenance it is important that all authorizations are checked regularly, but when there are many links and the rules are no longer transparent then it is hard to check. The survey shows that regular control of the granted authorizations is not done according to 34%.

Identity management is part of security management. The goal of security management is the accuracy, integrity and safety of all information system processes and resources (19). O'Brien and Marakas mention a collection of security measures that are necessary for security management in their book 'Management information systems':

- Virtual Private Networks, Firewalls, Network Security Protocols, Encryption, Access Control, Security Software Tools, Proxy Agents/Systems, Authentication and Intrusion detection.

In this thesis access control and authentication are the main subjects of discussion. Intrusion detection is also possible; administrators can keep logs of all access attempts and see if someone's access attempts are different from normal.

Security is often seen as something technical, but it can also be a business driver. For management to understand the vulnerabilities and benefits regarding security with identity management the following list might be useful. It includes vulnerabilities and identity management solutions for authentication, authorization and logging. These are three key security elements of identity management and they can help realize business benefits (8):

Authentication

Vulnerabilities without proper identity management:

- Bad passwords: passwords are too short or not difficult enough.
- Careless with passwords: because there are many systems and many passwords people write them down or lend them to other people.
- Social engineering: administrators or help desks (can) often not authenticate users in a reliable way which may result in giving unauthorized people access.
- Help desk misuse: at the help desk many people have access to passwords or can reset passwords.
- Accountability is difficult: password resets and changes in authorizations are often not logged, so audit logs are not present. If there is a security problem it cannot be traced back.

Benefits when using identity management:

- Stronger passwords: with identity management one can enforce stronger passwords. Users are required to choose longer more difficult passwords which may expire after a certain amount of time.
- One password: single sign on makes it easier for users to use multiple applications or systems without the need to login every time. With one password it is also easier to enforce stronger tokens, for instance smart cards.
- Help desk is less involved: with self-service (retrieve forgotten passwords etcetera) help desk does not need administrative rights and users can be asked for a combination of data or hardware tokens to authenticate before changing the password. The help desk is consulted less frequently and users can fix problems themselves which can reduce costs.
- Audit logs: identity management can keep logs of password resets, authorization rights etcetera.

Authorization

Vulnerabilities without proper identity management:

- Old accounts: when users leave the organizations accounts remain and when the role of a user changes within the organization his authorizations are not updated.
- Too much authorization: users can have too many rights because the policy is not correct. They can also belong to a group with too many rights, or maybe there is no policy and everybody has authorization for everything. This is contrary to the security policy of least privilege. The policy states that a user should only have access to the resources needed to fulfill his tasks.
- Conflicting roles: a user can have a combination of authorization that is unwanted within the organization.

Benefits when using identity management:

- Reliable accounts: because every user has one identity it is easy to maintain the accounts.
- Auditing: a privilege auditing system can be used to check the authorizations and modify them.
- Separation of duties: with a user provisioning system one can check for unwanted authorization combinations and remove them.

Audit logs

Vulnerabilities without proper identity management:

- Traceability: with many systems and accounts it is hard to link an action on one machine to another machine.
- Auditing: it is not easy to simply give a list of who has which authorizations. It is even more difficult to give a list of who had which authorizations at a given date.
- No logging is done: authorization change requests are not logged.
- Appropriate privileges: because the lack of a business process for identity management and authorization requests it is not possible to verify if the privileges are appropriate.

Benefits when using identity management:

- Traceability: with identification and logs it is possible to trace some action back to a user through various applications and systems.
- Auditing: a user provisioning system is used to request and authorize security changes. With that system a log can be maintained with all authorizations.
- Logging: the user provisioning logs changes so they can be checked later.
- Appropriate privileges: with a privilege audit system all management stakeholders are encouraged to review privileges and make a good decision about whether they are correct.

Identity management can help against lots of common security risks such as identity theft and security breaches from personnel. Unauthorized information access is in many cases not done by hackers, but by personnel. Also there are legislative and regulatory rules that the company has to follow. To implement those rules identity management and logging are essential. With identity management it is also easier to perform auditing.

Identity management is often beneficial for access control. It is shown that users tend to be more careful with their authentication information when access to more resources is possible with it. They tend to give passwords away less easily or let other people use their credentials. Safety can also be improved with logging and auditing all applications. Logging and auditing has another advantage it can possibly show identity theft (phishing). Also when using identity management it is necessary to explicitly formulate who has access to what. That can be done with for instance role based access control. When formulating access rights explicit and proper maintenance of those rights it is easier to see who has access to what. That makes it possible for management to become more involved, it is no longer just a technical system where the administrator gives permissions to users. The management can check the rights and set rules about the standard procedures for granting permissions.

3.2 Privacy protection

I think that identity management can be used to increase privacy as argued below. When users have lots of accounts and passwords, both users and administrators will handle the data with less care. But when that data is available to unauthorized users

then the privacy is an issue, since it is possible to access lots of information with the account name and password. Laws and legislation can also enforce privacy protection. For instance the WBP (Wet Bescherming Persoonsgegevens) in the Netherlands demands that users are aware of the purpose of registration. When there is only one place of registration instead of numerous places then that problem is reduced. Another advantage is that when you work with multiple organizations and a user from one organization needs access at another organization then with the help of federative identity management it is possible to just give the information from that user instead of all users. It is even possible to login anonymously, the system you are logging in trusts the system which knows your rights and just gives you the same rights. So with identity management your privacy can be better protected and access is still very flexible.

3.3 Risk management

The business benefits and the return on investment are not always considered. Some companies just use identity management as a precaution. They do not want to be held responsible for any privacy issues or sensitive data that has been misused. So when implementing identity management they know that they will not get extra costs in the future for law suits and other costs related to the risks they have not protected themselves against.

3.4 Regulatory compliance

Regulations such as Sarbanes-Oxley require stronger security, to protect sensitive business processes. Regulations such as Gramm-Leach-Bliley, HIPAA, PIPEDA and the EU Privacy Protection Directive 2002/58/EC require stronger security, to protect the privacy of investors, patients, consumers and citizens, respectively (8). Companies have to comply with those rules and they are forced to implement information security. It is necessary that only the people who need the information have access to it, which requires a very strict management processes. Unfortunately, many enterprises are trying to become compliant with separate and less efficient processes in order to ensure control (20). Identity management can be used to improve reliability, availability and confidentiality of information. It is easier to see who may access what or who has accessed something because it is centralized so management is easier. That can support audits and compliance to rules and laws.

For the compliance architecture there are some requirements for the implementation (8):

- Strong and reliable authentication.
- Effective controls over user access to systems and data, including automatic access termination.
- Audit trails that record user access rights across a heterogeneous environment, and over time.
- Periodic reviews of user rights, with integrated workflow to remove inappropriate access.
- Secure management of administrative credentials to workstations, servers and applications.

3.5 Operational efficiency

In many companies there are lots of applications, users need to create accounts and give different kinds of information. Often that data is entered differently in the various systems or people are even forced to enter the data differently. So every user has several identities, one for every system for which he or she needs to remember all relevant information such as username and password. The process to manage the

access rights for all those users to all those applications is difficult. When you want to give someone access to a couple of systems then you need to give that person access to every single one of them. When you want to block someone from using a system then you have to separately block that user from every single system.

With identity management it is possible to create one identity and use that one for all the various systems. That uses the SSO concept, when you have only one identity it is easier for the system administrator to maintain it. Using roles it is even possible to give someone rights based on the user's role within the organization. So instead of giving everyone separate rights you can put people in groups corresponding to their role and then give the group access rights. According to Quest (6) IT professionals have 4 sign-ons on average that they use daily. The users have to remember all those logins and they have to login for every separate application which costs valuable time.

Maintenance is made simpler: provisioning, maintaining and de-provisioning users and their information such as access rights can be done more efficiently. If an employee needs other rights because his role within the company has changed, then it is easy to give him or her access to the applications that are needed for the role. Partners and contractors can also externally access the resources that they need. With identity management it is possible to simplify and automate the processes necessary for access control and provide a central database with users and their rights.

New (online) services can be implemented faster. There is already an identity management infrastructure and processes for access management are already described. That makes it easy to setup identity management for new services.

3.6 User flexibility

In these days users want to access their information not from one fixed place, but from multiple places that can change over time. Those places can be inside the company or remote just through the internet. The users want to be able to access the resources they need around the clock. The information that those users need varies a lot, it is no longer possible to just simply divide the access to information in a few simple groups like 'users' and 'administrators'. Employees need specific rights for their tasks and those rights can change overnight. In the last years companies are trying to integrate systems with other companies which brings another problem: not only employees have to login, but also contractors, supplier's etcetera. To provide the flexibility that is needed to fulfill all those requirements it is a good idea to spend time on identity management. With identity management employees can access the information they need from various places in a secure manner.

3.7 User friendliness

Users can just login one time and access many different applications using SSO. With SSO users do not have to remember lots of credentials and can save time entering their credentials every time. Another advantage is that users do not have to contact various people to get access to all applications that they need, to change a password or to retrieve a forgotten password. With self-service they can change (some) information from their identity themselves.

3.8 Cost containment

The chances are slim that you will manage a cost reduction in a short period of time when starting to manage digital identities for employees. On the long run it is profitable to use centralized identity management instead of identities per application. When using centralized identity management the maintenance of the identities and

access control is easier. Changing access rights when someone gets a new role in the organization or adding new users is simpler. Also when building new applications you do not need to implement identity management for that application, because it is already managed by the central identity management. That saves costs for developing.

Companies do not always look at cost reduction when implementing identity management. Sometimes they implement it when they know that the chance is small that it will save money. For instance due to laws the privacy has to be protected, but that is not something that will reduce costs. However if the company does not implement security measures it can cost them much more because of law suits if the privacy is broken. Another way to reduce costs and make management more easily is self-service. With self-service users can maintain part of their data themselves and control their passwords. That reduces the work that administrators have to do and employees can save time as well. Retrieving forgotten passwords for instance can be done much faster.

At the IT department of the Umea University in Sweden they calculated that using identity management would save them 1 million euro's (14). In practice there is no explicit data that shows how much companies can save using identity management. But one expects that cost containment in the future is possible because of risks that are covered and cost reduction when developing new services.

When managing identities of costumers, cost reduction is more realistic. Internal organizations tend to adapt slowly to new systems, costumers adapt faster. But then it is still difficult to make those savings explicit. So it is hard to say how much a company can save, but one can assume that it will save costs.

3.9 Conclusion

In Figure 12 the reasons for identity management are shown again as it is important to realize them before the next chapter where we will introduce identity management within the organization.

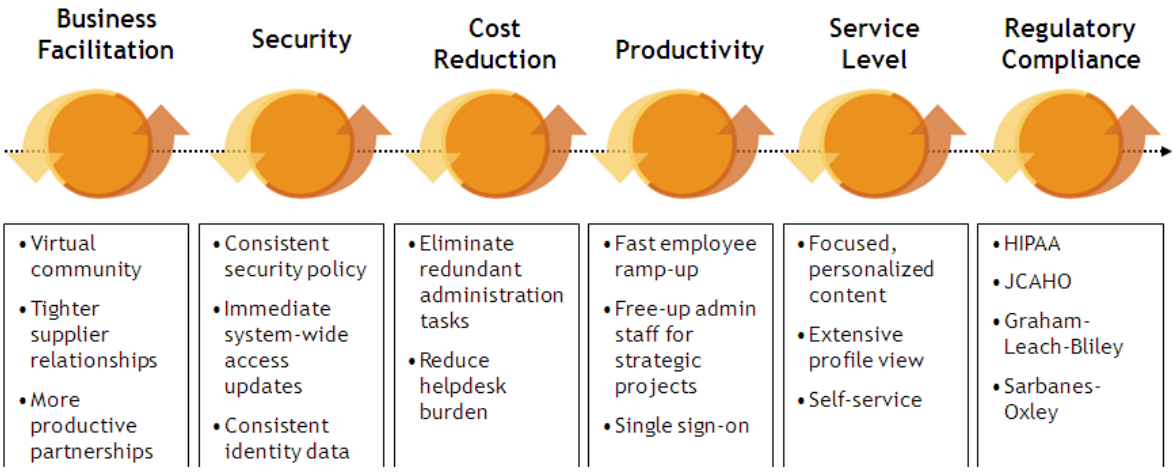


Figure 12 Identity management drivers[7]

[7] Source: Identity Management Strategy-Ensuring the Least Privilege Principle, by Cynthia Overby, Chris Bidleman for Novell.

These benefits can make life easier for users, administrators and management. Users are able to perform more tasks themselves and can get faster access. Administrators do no longer have to perform tasks that users can do themselves. The security is also improved which makes sure that administrators have less work and the privacy is better protected which is often required by law. Management sets the policies which are implemented by the administrators. The organization has the advantage that it is easier to work with other organizations such as suppliers, they conform to law and legislation and it possibly reduces costs.

The disadvantage is that implementing identity management is quite a job. All applications in the organizations should work with the identity management implementation. With the enormous number of applications and users in a company that might cost quite some time and money. Organizations are often reluctant to spend that time and money if they do not see the benefits. Hopefully those benefits are clearer after reading this chapter.

Which reason(s) the organization has to implement identity management can affect the rest of the implementation trajectory. Based on the reason(s) a certain level of identity management should be achieved. It is important that organizations know why they want identity management so that later they can check if all elements are realized. Also for most reasons you do not need to implement all identity management concepts, so knowing what you want can save time and money.

4 Identity management in a business environment

The previous chapter argued that the organization should establish reasons for identity management. When the organization knows why they want to introduce it then they have to figure out how they can do that. In chapter 10 some guidelines and possible consequences are introduced. This chapter treats some issues which companies will come across while implementing identity management. This chapter is intended as guidance for companies who want to introduce identity management in an effective way.

When organizations are small they mostly do not have an explicit authorization process. Everybody still knows each other and access to the various systems is gained by walking to the system administrator and requesting permission. When organizations grow the authorization process becomes more important. The number of different systems grows; the users have different passwords, different systems are maintained by different divisions and users need to wait longer for authorization. Then some important problems arise, because users have multiple passwords to remember they will choose simple ones to be able to remember them. Another problem is the fact that users have more authorizations then they need, after changing functions or leaving the organization the old rules are not updated.

For administrators it is not clear what the management policy is regarding authorization management, so they just give authorizations to employees who request it. The authorization rules are not maintained and become a big collection that is hard to maintain.

Also from a management perspective it is important to introduce a process of identity management. When there is no process with an authorization policy and managers do not understand the authorization rules as they are too technical then they will just authorize employees and that result in too many authorizations. An identity management process is not only necessary to make authorizations more transparent, but also to comply with rules and laws. Laws such as the Sarbanes-Oxley and code Tabaksblad in the Netherlands are meant to protect sensitive personal data and as a consequence companies should be able to provide a list of all authorizations and be able to support them. They might be able to generate these lists, but the lists are very technical and difficult to understand and support by management.

When introducing identity management it is important that it is seen as a business process and not just as a technical problem. Both the business or management perspective and the technical perspective should be regarded. A process and implementation should be constructed based on the administrative organization (AO) where the functions of the employees are eventually mapped to specific authorizations.

4.1 Administrative organization

Administrative organization is described by Jans (16) as: The complex organizational measures relating to data processing processes in an organization aimed at providing information for the controlling and operation of the organization and for making adjustments.

AO can be seen as the foundation for a 'good' organization as it provides guidelines for setting up or improving organizations.

Some of the issues AO deals with are:

- Risk management
- Compliance
- Transparency

We will see that those issues are also important for identity management.
With AO you define the processes which need to be implemented. An example of those processes is the internal control and security. Identity management can be used to implement some of the organizational measures that are necessary for an AO.

Another topic in the AO is data management. Identity management can be seen as a subset of data management. According to Jans (16) data management comprises the following:

- Define data definitions
- Define directives for the data use (or who may do what with which data)
- Check the execution of the previous activities

The last two items are also seen as the basics behind security which shows that AO and security are quite related.

Data management is becoming increasingly important due to data that is used by multiple persons. That poses some problems which are mainly security related. Due to the increasing importance of data management, the function is more and more deployed by someone upwards in the organization.

Changes in information systems need to be coordinated. Information systems are more and more integrated and more and more users use them so it is necessary to coordinate and monitor all changes. Identity management is necessary to control all access rights users have within an organization.

4.2 Causes of bad identity management

It is important to spot possible problems within an organization in an early stage. The same goes for identity management, some companies do not really need one; others do not know that it will benefit them. In this paragraph some causes or symptoms of 'bad' identity management are shown. When an organization has these symptoms then it might be time to think about identity management. Some organizations have no identity management whereas others have some ad-hoc implementations. Both can be seen as 'bad' identity management. But what are the causes for 'bad' identity management? Those causes can be divided into two parts; the technical and the organizational part with their relative causes, who can be found in (21), (22):

The technical causes are:

- Systems have different authorization mechanisms.
- A lot of systems were first bought or built for a specific division.
- The administration for the different systems is spread across the company.
- Security is seen as an add-on, first the base system is implemented and when that is ready they start to think about security.

The organizational causes are:

- The manager does not know what rights someone needs to fulfill their function within the company.
- There are no processes defined for requesting authorization and who needs to get contacted.

- There are no processes in place when someone changes function or leaves the company or something like that.
- Administrators have a lack of expertise regarding identity management.
- Awareness and training issues, users should be made aware and trained to understand the importance of identity management.

These causes combine to a mix of problems which makes it hard to maintain and improve authorization management. It is important to notice these causes as identity management should introduce some processes to solve those causes.

In the end the biggest cause for bad identity management is the increase in both number of applications and the users that need access to various resources. That has grown over the years, a couple of years ago one or two accounts where sufficient for most people. Nowadays companies work with many applications with their own mechanisms. Identity management can help to build one uniform logon for all those applications.

Another set of organizational causes of 'bad' identity management exist because identity management is not seen as a high priority issue. Other organizational processes are seen as more important and those are improved and optimized to increase business benefits. Or identity management is introduced ad-hoc because laws require it and it is not introduced to optimize business processes. The following points are given as reasons why identity management is not seen as high priority (21):

- The problem is spread into a number of minor sub problems
    - Users only experience the problem when starting or changing functions.
    - Costs are spread and hidden.
    - Users, managers and administrators experience different problems.
- Organizational problems
    - Ownership, no one feels responsible for identity management.
    - What kind of problem is it? An HRM, IT or management problem?
    - Who will pay for identity management?
- Technical complexity
    - It is a mess with a lot of authorization rules which make it not very transparent and as a result it is hard to improve.
    - There is no knowledge of possible solutions and tools.
- Psychological problems
    - It is not cool to clean the mess.
    - People find their way in the organization and they like it that way.

The Quest survey (6) made a list of the obstacles that have the most impact on the organization's ability to reach their identity management objectives. This is important as the organization should try to eliminate these obstacles to have a good foundation for identity management. A proper foundation is essential for the proper introduction of identity management so that it can fulfill all the benefits associated with it. The common obstacles that one should eliminate are (6):

Obstacles	%
Lack of funding	30,8
Insufficient staffing resources	18,8
Technological complexity	18,1
Lack of knowledge base to deal with technical issues	13,5
Lack of management support and direction	11,8
Other/ do not know / refused / none	7,0

4.3 Consequences of bad identity management

In companies it mostly takes some time before people realize the importance of authorization management. The problem is that the total impact within the company cannot be made explicit. Users are used to the problems with identity management and costs are spread over different divisions.

For companies it can be important to realize which (hidden) consequences 'bad' identity management has. It is necessary to make the problems with 'bad' identity management as explicit as possible so that organizations realize the importance of identity management. The following is a list of possible consequences for companies that do not have (proper) identity management (21):

- Passwords are not well chosen, they are mostly short and contain dictionary words, and also users write passwords on paper to remember them.
- Requesting authorization takes a while sometimes resulting in the fact that users borrow each other's passwords.
- Authorization rules are not up to date, people who change their function or leave the company still have access rights they do not need.
- Managers do not know which rights users have or need and just authorize without knowing what they exactly do.
- Managers do not understand the technical authorization rules so they do not understand the existing authorization rules.
- System administrators have an ad-hoc policy for granting access rights.
- Identity management is not a key topic and is treated late in the project. At the end of the project they swiftly determine who needs to get access which may result in problems like users who do not have the same amount of permissions that they had in the previous system.
- Higher costs for administration and the time it takes before users can become productive.
- Higher risks because of unauthorized access and possible sanctions for not complying with rules or legislation.
- Organizational issues when users become annoyed when waiting. If processes are vague or unclear then users can become unmotivated and possibly leave or start to act unprofessional because they think that the organization is also not professional.

Organizations need to know who has done what; therefore organizations need to know the identity of their users. As a consequence users have registrations for many different systems. These registrations have not much binding and are unreliable. Users use other users passwords, have bad passwords etcetera. That results in a big mess that is hard to maintain. Identity management is the solution for these problems.

With identity management you no longer have lots of authorization systems and you do not need to create a new account for a system when a user needs it. You have one identity which is maintained at one place. When someone starts working in the organization the identity is established and later changed when he changes function and ultimately removed when the employee leaves the company.

4.4 Business reasons for identity management

Organizations can have different motivations for implementing identity management. The kind of reason has an influence on the further identity management introduction.

In chapter 3 the motivations for identity management are explained, they can be summarized as follows:

- Compliance to rules or legislation.

- Usability, to use single sign on to reduce the number of logins.
- Cost reduction
- Maintenance improvement
- Improve security

These are reasons for why a company should use identity management and these reasons were further explained in the business drivers' chapter.

Identity management can have some 'side effect' advantages that are also very useful for an organization. Some advantages are:

- Possibility to use stronger tokens: when you have one login point you could use stronger tokens like smartcards. If you have lots of logins that is expensive and not really practical.
- Loose coupling between business and technology: with roles as connection between business and technology it is easier to change processes and keep the technology aligned.

4.5 Functional components

For an organization it can be interesting to see which functional components can be achieved with identity management. It is a bit technical but it shows some of the more technical benefits of using identity management. These functionalities can be a reason to use identity management or they can be a positive side effect of using identity management. What follows is a bit more technical list; the items below are used to realize the benefits of identity management. For management it can be good to see what functionality needs to be added to realize the benefits so they can see that it is not something simple. The items below are copied from one of the articles on identity management (8):

- **User provisioning:**
  - Automation / meta directory: used to propagate changes through various systems.
  - Self service workflow: enables users to request and authorize changes.
  - Consolidated administration: users from different systems can be managed from one central security division.
  - Delegated administration: certain users and systems can be managed by local security employees.
- **Privilege audit:**
  - Enable managers to review the authorizations of their employees and possibly maintain and clean up those rights.
  - Application owners can manage the list of users who have authorization to their system.
  - Group owners can manage and clean the list of users who are part of their group.
- **Password / authentication factor management:**
  - Synchronize passwords between systems in the network.
  - Enable self-service, for instance with assisted password reset possibilities.
  - Enroll and maintain profiles of challenge-response data for users.
  - Enroll and maintain strong authentication factors, such as biometric samples or hardware tokens.
- **Single / reduced sign-on:**
  - As mentioned before SSO reduces the number of times that a user has to enter credentials to access various systems.
- **Reporting:**

    - Generate a matrix which connects users to resources and privileges.
    - Generate reports that contain a list of requests and authorized changes including the time they were introduced.
    - Identify anomalies, such as orphan and dormant accounts.
- **Virtual directory:**
    - Consolidating multiple and possibly heterogeneous directories into a single, global view.

4.6 Risk analysis

When companies think about implementing security measures they mostly perform risk analysis. When the risk is low then maybe it is too expensive to implement the security measure. But for enhanced security, companies often implement it even if the risk is low, just because if it happens the impact can be big. Laws and legislation can also require that companies have appropriate security measures. When doubting about identity management it is a good idea to do risk analysis to see what risks your company has without identity management. Sometimes companies even implement elements if the risk is low, just to avoid possible consequences which is also known as risk management what was explained in 3.3.

Risk analysis can also be found in the administrative organization as one of the items in AO (16) is the internal control and security. Security is necessary to provide quality as one of the requirements to information technology is that it should be available all the time and it should be correct.

There are some categories of security measures:

- Physical
- Organizational
- Hardware
- Software

This thesis is mainly about software security and the organizational aspects that are necessary to provide that security.

Risk analysis is described (12) as systematically recognize and evaluate measures against possible unwanted events in such a manner that conclusions can be drawn from it. An organization should do risk analysis to see which areas need extra attention and that could lead to the implementation of identity management to cover those risks.

When performing risk analysis the company goes through the following steps:

- Describe possible risks
- Determine the probability that each of those risks occur
- Determine the damage if a risk occurs

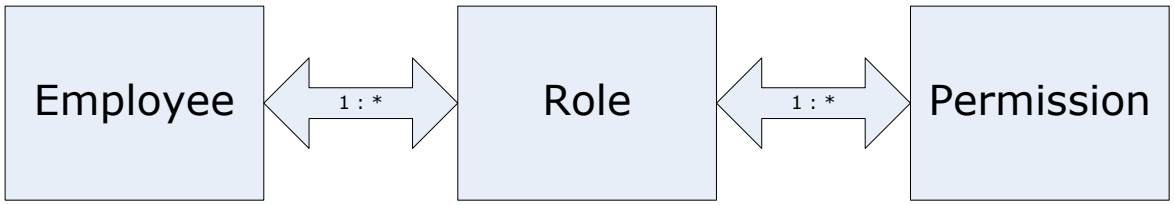
After the risk analysis one determines which measures will be taken to prevent, signal, repress or correct risks and what the costs will be. If an organization does not see the importance of identity management then risk analysis might show some risks that can be covered by identity management. And as mentioned before, risk reduction can be a business driver to implement identity management.

4.7 Coupling business and technology

As mentioned before the identity management issue is not something technical. Identity management should be part of your business. Then the (identity

management) business processes can be supported by technology. This paragraph shows how business and technology can be linked together.

For the actual implementation of the authorization roles most organizations use the Role Based Access Control (RBAC) standard. Authorizations in RBAC are not directly coupled to persons but roles are put between them. One employee can have multiple roles such as a secretary and a financial role. Every role has one or more permissions that are needed to fulfill the role. For the financial role someone should have access to the application where the finances are administered and most people need permission to access printers etcetera.



As always in security it is important to follow the principle of least privilege, or to give everyone as many authorizations as they need but nothing more. That proves difficult with identity management as all minor subtasks need to be described in detail which might be too much. Therefore it is important to make some kind of risk analysis and decide if it is a big risk if someone gets just a bit too many authorizations.

The advantage of RBAC is that if you have multiple persons which you all want to give the same authorizations the maintenance is easier. Normally with 20 persons and 7 authorizations per person you would need to maintain 20*7 links. When you add one role you would need to maintain 20 links from the person to the role and 7 links from the role to the authorization. Roles can be grouped together to form a new layer of roles.

Another possibility is to make a hierarchical model. Then you would have something like the role CEO where the role management is beneath him and beneath the management role you would have the employee role. That means that the top-role has all the permissions from the underlying roles. This has the advantage that you do not need to couple the same authorizations to different roles, but can make it less transparent. It is most useful in a big organization with a lot of the same roles. As this method is not often used in practice we will only consider RBAC in the rest of this thesis. For more detailed information about RBAC and the more advanced features it is best to look at the 'Proposed NIST Standard for Role-Based Access Control' (23).

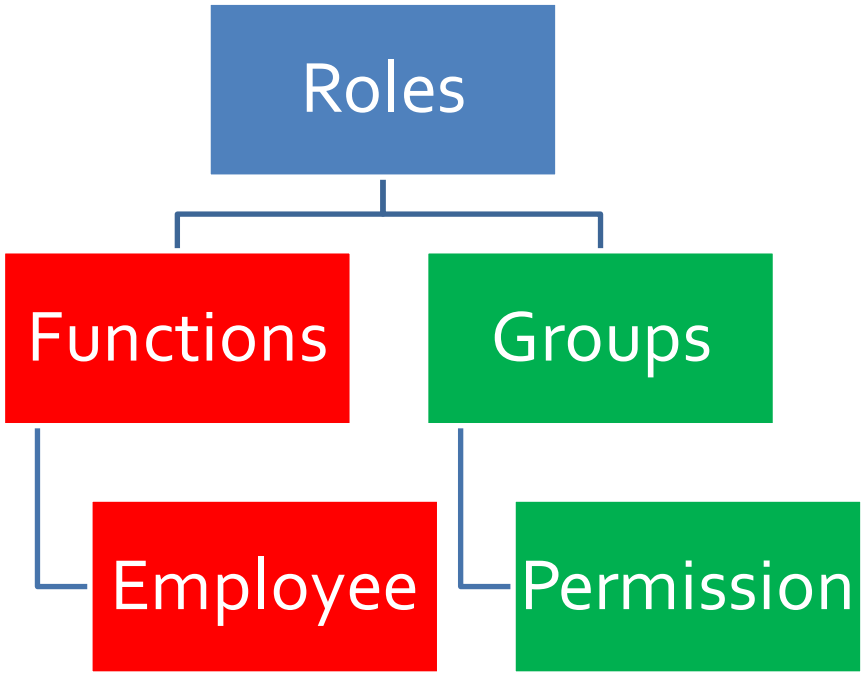
Another choice that needs to be made is if you want a centralized or a decentralized system structure. With a centralized structure you have one database which has the advantage that authorizations are always up to date and corruption is minimized. The disadvantage is that you have a single point of failure. If something happens to the database users can no longer access the resources they need. With a decentralized structure you have one central database and a read-only (only the central database can modify it) copy for every application. The advantage is that there is no single point of failure which benefits the availability. The disadvantage is that the copies may not be up to date as it takes some time for changes to propagate through the system or the copies are corrupt. These issues become less of a problem due to the improved software and infrastructure which lowers the risk for such problems. Maintenance and costs of maintaining two different systems might be some important disadvantages. A possible influence for the choice between centralized and decentralized might be the infrastructure. If multiple platforms (Windows/Unix) are used then a decentralized

structure is more common. With a single platform a centralized structure is mostly enough.

Other than RBAC some other additions are used within organizations:

- *Rule* Based Access Control: control based on rules or policies, for instance that the employee can only access resources from within the company.
- Function separation: to make sure that a user cannot have two sets of authorization at the same time where the combination of authorization is unwanted. Function separation is also known as 'constraints'.
- Auditing: necessary for rules and legislation compliance, the choice and type of auditing has a big influence on the choice of product and the system.
- Self service: for instance when forgetting a password or for managers to give rights to employees themselves. That reduces administrators and helpdesk time which reduces costs.
- Single sign on (SSO): an employee has one login and performs that login only ones and can access all systems he has authorization for. It is also possible to setup SSO first and then introduce IM so that users are not aware of any differences.

To see the connection between the business and the technical part the following picture might be illustrative. The picture shows how roles can be used to bind the business and the technical part. An employee has some functions these functions can be seen as roles that the employee has in the organization. The employee needs authorizations/permissions to be able to fulfill the function within the organization. Permissions are bundled in groups and those groups can be seen as the technical roles of the organization. The technical and business part is thereby coupled by roles as was also shown in the previous picture where the one too many relation between employee, role and permission was shown.



In the picture above the red elements on the left show the business part and the green elements on the right show the technical part. Without identity management you would have many bindings between them, but with identity management you just bind them through roles. When using roles the management team can develop a

process and the administrator can focus on a role based solution to support that process. That way the management can give authorization based on the functions of the employees within the organization.

The usage of an identity management system within a company process is part of the Administrative Organisation (Administratieve Organisatie - AO). Remember that identity management supports the company process; identity management is not just a technical solution on itself.

After coupling users to roles it is important to know what you want to protect. Maybe some resources do not need any protecting or some users should have access to everything. To know what you want to protect you could use the Four W's (22):

- Which applications are we protecting?
- Who are we protecting the applications from?
- Where should we protect them?
- Why are we protecting them?

With identity management you should see applications as resources since identity management is not only about applications but also about systems and other resources. Then you first have to determine what resources you want to protect, you could do that with risk analysis which is treated later. After you know what resources you want to protect you should determine who has or should have access. Do you want to allow for instance partners, suppliers and external users? That could complicate identity management as you would need somewhat more advanced features such as federated identity which is also discussed later. Then you should determine where you want to protect them, do you simply disallow access in the firewall for all external users or do you control access in your application. The last important step is to decide why you protect the resources. Damage to the resources can have serious consequences for your business. Administrators have more work and users cannot perform their usual tasks. Another important step is to setup recovery processes for the resources in case something goes wrong.

4.8 Implementation issues

It is important to implement identity management in steps. That way the benefits can be seen at an early stage and one can build upon earlier successful implementations. There are many maturity models for software development and things like that. For identity management the companies that offer complete identity management solutions have their own maturity models for identity management.

After determining how you want to set up or change identity management the actual implementation is the following big step. According to CA (24) there are five keys to a successful identity management implementation:

1. Know Where You're Going: you need to know where your current business is and how the security is, and then it is important to have a business perspective and connect the phases of the identity management project to business results.
2. Get the Right People Involved: people with different professions need to collaborate. IT personal, management and the owners need to work closely together to improve their jobs.
3. Implement Incrementally: incremental implementation shortens the 'time to value' of the project and one can build further upon a successful implementation.
4. Educate, Educate, Educate: the end users and the IT personnel need to keep up with new developments. Vary the kind of training as that makes it easier for people to remember.

5. The Job is Never Done: the identity management system needs maintenance with product updates and changes in the IT environment or the organizational environment.

These steps should make sure that you can realize value from investment quite soon and then can build up to add to that value. Value is constantly added to the organization and you can educate personnel so they realize the value of identity management. That makes sure that management can see the value of further improving the identity management.

In this paragraph two maturity models are introduced, the first one from CA focuses on the business value. It shows the business value that can be achieved if the organization reaches a certain level of maturity. That list is probably most useful to management as they can see what benefits the company can have at which stage. Management can use the model to determine which level of maturity they want to achieve the business benefits they want. The second model is by Oracle and is more focused on the functional components. This model is probably more interesting for the more technical people involved in identity management. The model can be used to see which functional components are necessary to achieve a certain level of maturity. When management has decided which level of maturity they want then it is possible to see which functional components should be implemented to achieve that level. Note that the levels differ between the models, but both models start with low level, simple identity management and end with the more advanced identity management solutions.

CA which is a big identity management company with 3.94 billion USD of revenue in 2007 uses an identity management maturity model (25) to estimate at which level the identity management process capability is now and they use it to help build towards a blueprint for a solution. It contains 4 phases.

1. Active: Integrated Credential Management
   Business value:
   - Enhanced user productivity
   - Reduced helpdesk costs
2. Efficient: Enterprise Identity Provisioning and User Access
   Business value:
   - Accelerated business applications
   - Reduced cost of user provisioning
3. Responsive: Access Control for the Extended Enterprise
   Business value:
   - Reduced cycle times for new hires
   - Reduced cost of compliance and partner management
4. Business driven: IAM Integration with Enterprise Risk and Asset Management
   Business value:
   - Reduced cycle times for IT Services
   - Reduced cost of integrating with risk and asset management

Oracle offers their own model (26) to measure the maturity of identity management where the levels are measured by looking at the already implemented identity management elements. The model separates the following three levels of maturity:

1. Tactical
   - Web Access Management
   - Enterprise Directory
   - Password Management
   - Meta Directory
2. Process-Centric
   - Enterprise SSO

- Automated Provisioning
- Consolidated Reports
- Virtual Directory
- Enterprise Roles

3. Aligned
    - Full Regulatory Compliance
    - Converged IT & Physical Security
    - Identity Federation
    - Risk Management

As said before, these maturity models can be used to see where identity management is within your organization, what can be improved and assist in the blueprint. The maturity models do not only show which functionality is missing, but also which business benefits one can achieve when maturing identity management.

4.9 Implementation scenarios

In the paper by Bosch (21) some different scenarios are mentioned to develop an identity management solution. It is of course important to realize why you want to introduce identity management, but you should also determine from which point of view you want to start. Do you want to implement a solution, do you want to implement a process or something different. The different scenarios from the paper are mentioned below.

This paragraph relates to the 'Know where you are going' concept which was introduced in the previous paragraph. It also closely relates to the guidelines chapter where the steps to a successful implementation of identity management are covered. It is important to realize early in the project where you are going. The management should make a decision about the further implementation of identity management and build further on that decision.

Aimed on the process:
Central is the deployment of identity management and RBAC based on existing company processes. The AO is used to setup RBAC. In practice a role model is created for every system, where a system can use roles from previous systems in new systems.

Aimed at the platform:
An existing identity management system is used to control other systems. For instance Active Directory can be used to manage all kinds of systems.

Aimed at a solution:
An existing solution for IM is chosen. This solution is used as IM for all systems.

Outsourcing:
Identity management processes are outsourced, for instance identity management (in this case only identity management is meant, not identity and access management) is outsourced and used for the authorization system in-house.

Creation:
Create an identity management solution based on the company processes with the help of RBAC. From the AO a role model is constructed which is implemented in a new identity management system.

Federation:
An organization uses the authentication mechanism of another organization and trusts that organization. So an organization sends the request from a user to another organization and uses the identity management from that organization. For instance with the GSM network, if you try to connect through a different provider then your own provider then that provider gives you access based on the information it gets from your provider.

At the beginning of the path to identity management the organization should choose a scenario and stick with that. It is important that the decision about the scenario is made early in the process as it can result in the purchase of software or the build of software which might take some time and involves management and administrators.

4.10 Access control issues

There are some conflicting targets with identity management: identity management should be flexible and maintainable and on the other side it should be secure. Those issues can be conflicting. That authorization is an issue which should not be neglected is underlined by the survey from KPMG (9) which shows that many users have more rights than they need.

The security target of access control is to ensure things like exclusiveness, continuity and integrity (12). To make those security targets possible the authorization should be very restrictive. Users should only be granted access to the specific resources they need and nothing more. This is also known as the Principle of Least Privilege which states that users should be given no more privileges then necessary to perform their job. The original formulation from Saltzer and Schroeder dates from 1975: 'Every program and every user of the system should operate using the least set of privileges necessary to complete the job.'(27) That it was not only an issue in the past but also in the present is shown by recent formulations by for instance Bishop: 'The Principle of Least Privilege states that a subject should be given only those privileges needed for it to complete its task.'(28)

The principle of least privilege should result in several security measures:

- Eliminating unused user accounts.
- Let users change default passwords.
- Limit resources access to a minimum set of privileges.
- Limit resources to when they need them, administrators should not use their admin account for daily use.

The benefits of implementing the principle of least privilege include:

- Compliance to law and legislation.
- Increase productivity: there will be less security issues which benefits administrators, there is less downtime which benefits user productivity.
- Prevent unauthorized use: users cannot access resources they do not need; they cannot modify data they are not authorized for which minimizes errors.
- Increased security: administrators do not work with admin accounts when they do not need them.
- Reduced costs: less security problems which increases productivity and reduces costs for downtime.

So from a security perspective you want to set very strict authorizations giving users only the rights they need at that moment. From an organization perspective that can be both good and bad. Compliance, reduced costs and security are wishful, but

restricting users can prove to be not beneficial for the organization. So the security and organizational perspective can be conflicting. To see what an organization wants to do with information (systems) we will look at the Administrative Organization and Bestuurlijke Informatiesystemen. These are important topics from the organizational theory that deal with the construction and control of organizations. I will compare the concepts behind security and the organization to show the different ideas.

The Administrative Organization is described by Jans as follows (16): 'The complex of organizational measures in an organization that focuses on the information provisioning'. Where information delivery can be seen as providing the information needs of employees in an organization so that they can fulfill their role within the organization. As well as the provisioning of information according to the needs of external stakeholders or other interested parties.

In 'Bestuurlijke informatiesystemen en automatisering' (15) management requirements of systems are treated. According to the book systems should be flexible, maintainable, testable and transferrable.

That results in an organizational perspective where you want to support users to do their job. Therefore all the information they need should be available. That can be conflicting with the principle of least privilege where you want to restrict the access to resources as much as possible. That can result in users having to request authorization on a regular basis as they have to do something they are not authorized for. That will annoy users and it costs time for users and administrators to regulate those changes.

So there are two problems with very strict authorization management:

- The time and price it costs to implement very strict authorization rules. The organization has to find out exactly what rights every user needs and all those rules should be implemented by the administrators. It is very complicated to implement everything in a way that benefits both the organization and the security.
- The conflicting interests, from an organizational perspective you want employees to work as best as they can and increase their productivity. Implementing the principle of least privilege might hinder users when they want to do something they are not authorized for.

The second problem can be solved to some extent. When the user authorizations are made very careful such that users have access to exactly the resources they need to perform their job then that helps already. But it will take quite some time from both management and administrators to implement it successful. The problem however is that it is often difficult to forecast the rights someone needs as functions change and the need for specific systems or other resources changes. Only with standard jobs where someone performs the same actions for years it can be implemented correctly, but most jobs have changing needs for information. Another issue is that the principle of least privilege states that you should give someone only access to resources at the time that the user needs it. So when a user at sometime does not need a resource then it should be blocked for that user. That means that users have to requests resources if they need them and when they do not need them for some time then they should not have any access. That is very hard to automate if not impossible. So if the principle of least privilege is implemented fully it will probably mean that users have to request for resources on a regular basis. That is not what you want from an organizational perspective as it costs time for administration and is annoying for users.

This results in conflicting interests; one wants a high security level while keeping the system flexible and maintainable. The problem is that authorization rules should be very specific when you want to comply with the Least Privilege statement which results in lots of rules and that makes the system not very flexible and hard to maintain for the administrators. For the users it can be problematic, rules are very strict so if a user wants to access something he normally does not use then he probably has no permission. This will lead to lots of permission requests from users and administrators have to maintain lots of access rules. The business will not benefit from too strict identity management, administrators and users will not like it and the costs are very high.

It might be an option to implement the privilege of least privilege to some extent. So when implementing access control one has to evaluate how strict the security should be. For some sensitive data it can be important to use strict policies while some public information might not even need access control. One should make sure that users can perform their tasks and not restrict them to much only to provide service for some exceptional cases that rarely occur. To estimate when to implement tighter security or not the organization can perform a risk analysis on all resources.

A related issue is how you want to setup your access control. Do you want to deny all access except when someone is authorized, or do you want to allow all access except for some protected resources. With identity management you usually deny all access and give users authorization for the resources they need. For flexibility and maintainability it might be better to allow all access except for some protected resources. This implies the same problems as stated above; the organization has another perspective then the ideal security perspective. Organizations should find a balance between security and flexibility in order to be as successful as possible.

Another issue is the employer, administrator or manager issue. Do you want to give the employer, administrator or (some) managers access to all resources? Especially for privacy that might be an issue but for employees it might be hard to check who has access to what resources. One should look very careful to this, you want administrators to have lots of access to assist users but you do not want them to be able to read users passwords for instance. Who has access to what should be carefully defined when implementing identity management.

All these questions should be answered explicitly before introducing identity management as they are essential for the further development of the identity management process.

4.11 Conclusion

In this chapter the causes and consequences of identity management are described. It is important to see what can result in 'bad' identity management and what the consequences are. For organizations it is important to introduce identity management as successful as possible because that will benefit the organization most. This chapter also treated the coupling between the business and the technical part of identity management so that both sides know how it works together. Another important aspect of this chapter was to introduce some a bit more technical concepts of identity management for the management so that they understand a bit more of the technical side. The end of this chapter was dedicated to maturity models and access control issues. For management it is important to set the targets of identity management so that one can check later if those targets are achieved. The access control part shows that both security and the organizational have some targets in common, but there are also some conflicts between the two. When implementing identity management one

should consider these conflicts and see what is best for the organization and put that on paper. For the implementation of identity management there is a separate chapter about guidelines, that is chapter 10.

5 Directory servers

In the previous chapters the business part of this thesis was treated. The reasons why one should use identity management and how it should be introduced where explained. In the next chapters the more technical part is covered. This part shows how the actual implementation with the help of software can be done to realize the benefits of identity management.

This chapter is about directory servers, most often they are used to verify a user. If there is just one computer logging in is done locally and the username and password are stored on that machine. If there is a network with printers etcetera the usernames and passwords are mostly stored on a server. When a user is logging in the server checks the username and password to see what the user is allowed to do. Some users might have access to printers while others do not have that access.

A directory server can be seen as a dictionary (or telephone book) where it is possible to lookup the (user) name and retrieve al information regarding that name (also known as the identity). Unlike a dictionary a directory server is not sorted alphabetically but in a hierarchical tree.

In this chapter some important concepts of directory servers are explained. After that the configuration I used to test my 'identity management' examples is discussed including the directory servers I used. I chose some directory servers to explain them in more detail and discuss the experiences I had when using them. The final part consists of a comparison of the different directory servers I used.

5.1 Important concepts

Directory servers are mainly based on the Lightweight Directory Access Protocol. Domain Name System is used to make sure that the computers in the network can find the directory server. Kerberos is often used as authentication protocol, for Active Directory it is the standard protocol. These concepts are probably new, so if you want to know the basics read this paragraph, if you are familiar with these concepts you might want to skip this paragraph.

5.1.1 Lightweight Directory Access Protocol

The Lightweight Directory Access Protocol (LDAP) is based on the X.500 Directory Access Protocol (DAP), but uses TCP/IP instead of the Open Systems Interconnection (OSI) protocol stack. LDAP was intended as a lightweight protocol for accessing X.500 servers by using the TCP/IP protocol stack. LDAP is a network protocol that describes how the information from the directory server should be retrieved over for instance TCP/IP. If one sees the directory as a telephone book, LDAP is a description about how to retrieve the data the user wants from the telephone book. LDAP is just a description; OpenLDAP and Active Directory are examples of an implementation of LDAP.

LDAP consists of a tree and can be defined in the following way:

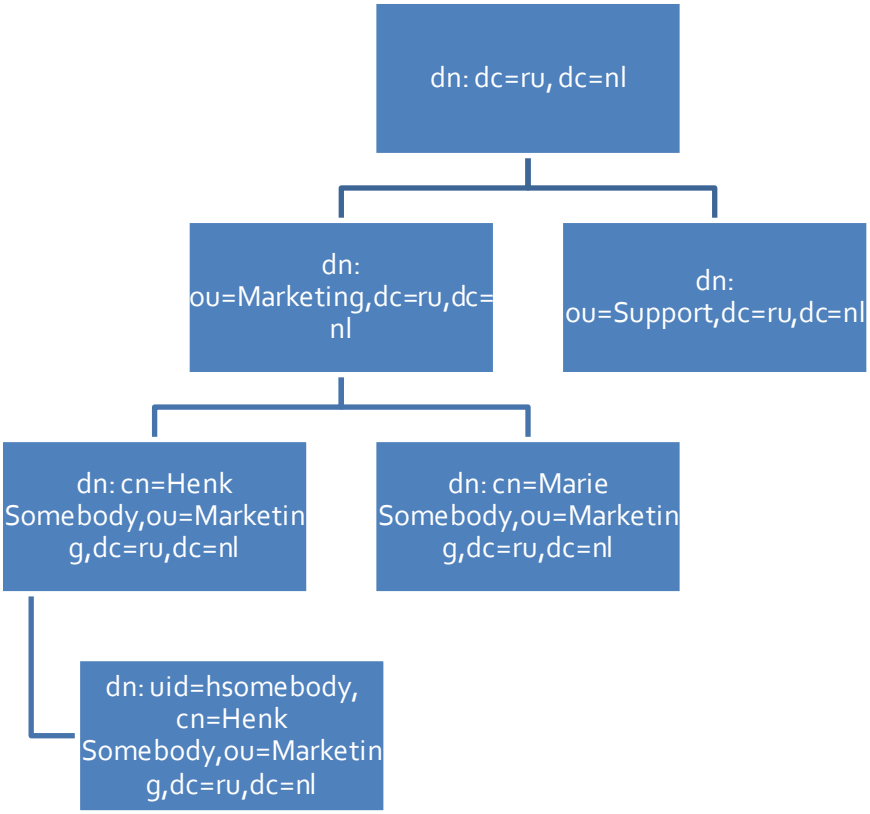
- A tree contains directory entries
- A entry contains some attributes
- A attribute has a name and some values

Each entry has a unique identifier, the Distinguished Name (DN). The Distinguished name can be seen as the full URL of a website. The DN can be used to find the entry.

The following attribute names are used in the example:

- dc: Domain component
- ou: Organizational unit
- cn: Common name
- uid: Userid
- dn: Distinguished name

What follows is an example of a LDAP tree:



LDAP is widely used and organizations have built their own implementations. Unfortunately they are not fully compatible to each other. So when using different LDAP implementations you have to make sure that they can work together. Some commonly used server versions of LDAP include:

- Active Directory
- OpenLDAP
- Fedora Directory Server
- IBM Tivoli Directory Server

Active Directory is the Microsoft implementation of LDAP. OpenLDAP is a common used Open Source implementation of LDAP. Both will be used in this thesis to test the prototypes.

Active Directory Application Mode (ADAM) is worth mentioning as it might be a new step for directory services. ADAM is a lightweight implementation of AD and runs as a user service and not as a system service such as AD. The advantage of ADAM is that it

provides all AD functionality, but you do not have to create domains or domain controllers. That can save costs when setting up an infrastructure.

5.1.2 Domain Name System

Domain Name System (DNS) is a protocol to translate domain names to IP-addresses and to translate IP-addresses to domain names. For instance www.ru.nl corresponds to 131.174.93.174. For users it is easier to remember domain names, so it is necessary to translate between the two. In this project the DNS server is necessary for the clients to find the LDAP server.

5.1.3 Kerberos

Kerberos is an authentication protocol which users can use to prove their identity to one another over an unsecure connection. This is done by issuing tickets that can be used to access a resource during the active session. Active Directory uses it together with DNS to provide an identity management system.

Kerberos is based on three items: the client (user), the service and the Key Distribution Center (KDC). The KDC contains an Authentication Server (AS) and a Ticket Granting Server (TGS).

The authentication process to an LDAP server can be described in the following steps:

- The client sends his username and password to the Authentication Service (AS) to authenticate.
- The AS sends the Ticket Granting Ticket (TGT) and the session key to the client.
- The client sends a request to the Ticket Granting Server (TGS) containing the TGT and an authenticator generated by the client.
- The TGS sends a service ticket and a service session key.
- The client sends a request to the LDAP server with the service ticket and an authenticator generated by the client.
- The server replies to prove his identity.

5.2 Configuration

In this paragraph the actual configuration of my virtual servers is explained, it is not one of the most interesting topics. For future reference it might come in handy as it shows on which versions of software my research is constructed.

For my research I mainly used a Windows XP system with Virtual PC 2007 on it. In Virtual PC I created a local only network. In that network I used the following virtual machines:

- Ubuntu Server 7.10 with OpenLDAP and DNS server (BIND)
- Windows server 2003 with Active Directory, Kerberos and DNS
- Windows XP used for building and testing the prototypes

Both servers used IP address 192.168.1.200 and the directories contained a test user and password.

For the development and tests I needed a virtual machine inside my local only network. Therefore I used a Windows XP machine with the following software:

- Visual Studio 2005
- Eclipse 3.3.1.1
- Tomcat 6.0

The client machine had IP 192.168.1.201 and was put in the domain *domain.com*. With a test user and password I tried to connect to the servers.

A separate VMware virtual machine was used to test Fedora Directory Server.

5.2.1 Windows Server 2003 with Active Directory

Active Directory is an implementation of LDAP which is targeted to Windows users (both end users and Windows administrators). With Active Directory you do not only have a place to centralize authentication and authorization, you can also deploy software and critical updates within the organization as well as other features.

The following steps are necessary to setup an Active Directory server:

1. Install Windows server (in my case 2003)
2. Configure a static IP address (191.168.1.200)
3. Change the computer name: host.domain.com
4. Install DNS: Add/Remove Windows components, DNS, use 192.168.1.200
   Configure DNS: create a Forward Lookup Zone with domain.com
   Create a Reverse Lookup zone with 192.1681
5. Configure domain controller: run dcpromo
   - Follow the steps

For a more detailed guide you could follow the How-to Install Active Directory on Windows 2003[8] guide.

If you understand the basics of DNS and follow a guide then it is quite easy to setup an Active Directory server. If you have any problems then there is quite some documentation.

5.2.2 Ubuntu Server 7.10 with OpenLDAP

OpenLDAP is an open source implementation of LDAP which runs on multiple platforms. The main advantages of OpenLDAP over Active Directory is the open source implementation which means the software is available for free and can be modified. Another big advantage is that OpenLDAP is not platform dependant.

The following steps are from a guide[9] and are necessary to setup an OpenLDAP server:

1. Install Ubuntu (in my case Server 7.10)
2. Enable all repositories
3. Configure a static IP address (192.168.1.200)
4. Make sure the server can be found: edit /etc/hosts change into the following *127.0.0.1 host host.domain.com*
5. Install OpenSSH Server: apt-get install openssh-server
6. Install OpenLDAP: apt-get install slapd ldap-utils migrationtools
7. Configure OpenLDAP: dpkg-reconfigure slapd (use domain.com)
8. Restart OpenLDAP: /etc/init.d/slapd restart
9. For configuring Samba, LDAP authentication and the DNS follow the rest of the guide.

[8] Active Directory guide:
http://www.petri.co.il/how_to_install_active_directory_on_windows_2003.htm

[9] OpenLDAP guide:
http://www.howtoforge.com/openldap-samba-domain-controller-ubuntu7.10

For installing OpenLDAP it is good if you understand DNS and have some experience with UNIX. With the available guide it is quite easy to install a basic LDAP server. The problem is that when something goes wrong then the available documentation may fall short and configuration might be difficult. That was also the case when I tried to add Kerberos support, something broke and there was not much (useful) documentation around to fix the problem.

5.2.3 Fedora Directory Server

Both Fedora Directory Server and OpenLDAP are derivatives of the slapd project. The developers of the slapd project became employees of Netscape where they developed the Netscape Directory Server which resulted in the Fedora Directory Server. OpenLDAP and Fedora Directory Server are both based on LDAPv3 and have comparable features.

OpenLDAP is more console based which makes it difficult to compare with Active Directory. A graphical user interface mostly benefits the usability as well as other software characteristics. Therefore I also looked at Fedora Directory Server which can be controlled with a Java application or web based[10].

To run Fedora Directory Server on my Windows XP machine I used VMware because installing Fedora in Virtual PC gave some problems. The installer stopped, or the installation became corrupt after installing the Directory Server and running an update.

The following steps are necessary to setup a Fedora Directory Server:

1. Install Fedora Core (in my case version 8)
2. Disable SELINUX: change *SELINUX=enforcing* to *SELINUX=permissive* in */etc/selinux/config*
3. Make sure the following ports are opened in the firewall: 389 (ldap), 636 (ldaps),9830 (admin Fedora Directory Server)
4. Make sure the server can be found: change */etc/hosts* with the following line *127.0.0.1 host.domain.com host localhost.localdomain localhost host*
5. Change the hostname: use your Fully Qualified Domain Name (FQDN) (*host.domain.com*) in */etc/sysconfig/network*
6. Update fedora: *yum update*
7. Add repositories for Directory Server: *cd /etc/yum.repos.d*
   wget http://directory.fedoraproject.org/sources/idmcommon.repo
   wget http://directory.fedoraproject.org/sources/dirsrv.repo
8. Install Directory Server: *yum install fedora-ds*
9. Run the installation script: */usr/sbin/setup-ds-admin.pl*
   - Use the FQDN as computer name
   - For the administration domain and directory server identifier use: *host*
   - Use the following suffix: *dc=host,dc=domain,dc=com*
   - For the rest you can use the defaults

Then you can use the Java application by executing */usr/bin/fedora-idm-console* or the web interface by browsing to *http://localhost:9830*.

If Kerberos is needed then you can follow a guide on the Fedora Directory Server site and set it up.

[10] For screenshots see the official Fedora Directory Server site: http://directory.fedoraproject.org/wiki/Screenshots

Fedora Directory Server was easier to setup than OpenLDAP and it provides a nice interface which improves the usability. So with the available documentation and some UNIX knowledge you should be able to set it up yourself. The project itself has quite some 'extra' documentation if you want to add things like for instance Kerberos.

5.3 Other directory servers

The above directory servers are certainly not the only ones. I choose Active Directory as it is one of the most used directory servers. That is underlined by the GSS 2004 IT Survey[11] which shows that 37% uses Active Directory, 19% uses Windows NT Domain and the rest is quite spread. OpenLDAP and Fedora Directory server I choose as I wanted to compare Active directory to something Open Source.

Maybe some other directory server might be better for an organization but it goes a bit far to compare all of them and the market changes over the years. A research by IDC[12] showed that 80 percent of the Fortune 1000 companies use eDirectory. But the Yankee Group Consulting White Paper by Laura DiDio 'Legacy Netware/eDirectoru Customers Flock to Windows Server 2003/Active directory'[13] showed that by 2004 only 11% of the businesses used eDirectory. So it seems that this directory server has lost quite some market share over the years. That shows that it is not a stable market, over the years there where different popular directory servers.

If someone wants to research more directory servers, then the following list of other popular directory servers might be useful:

- Apache Directory Server
- Red Hat Directory Server (Commercial version of Fedora)
- Sun Directory Server
- Oracle Internet Directory
- Apple's Open Directory
- IBM Tivoli Directory Server
- Novell eDirectory

5.4 Directory server comparison

Above three different implementations of LDAP are described. But which one is the best pick for an organization? The differences of the directory servers are roughly described below.

To compare the three directory servers are compared using the quality aspects discussed in 2.12. The comparison will be based on the ISO 9126 standard.

Functionality

- OpenLDAP
  - \+ Implemented according to the LDAP standard.
  - \- Only standard functionality.
  - \- No web interfaces.
- FDS
  - \+ Implemented according to the LDAP standard.
  - \+ Extra functionality.
  - \+ Web interface.
- AD

[11] http://www.gssnet.com/research/GSS%20IT%20Survey%20Report%202004-final.pdf
[12] http://www.novell.com/products/edirectory/whychoose.html
[13] http://hosteddocs.ittoolbox.com/Migrating_from_NDS_to_Active_Directory.pdf

- Implemented according to the LDAP standard, but with slight modifications which makes it harder to use applications that follow the standard very strict. But because of the fact that Active Directory is more or less the standard, most programming languages/frameworks etcetera have support for Active Directory. But this still affects portability.
+ Extra functionality but some is only available for Windows clients like deploying software through the network.
- No web interface.
+ More third party add-ons.

The interfaces can improve some software characteristics. Using an interface instead of commands is easier to use and more time efficient and improves the maintainability. Administrators are supported by the interfaces to improve the quality of their work.

Reliability

- All three directory servers are very reliable; they exist for some years and are improved over the years. There is some debate about Active Directory or the Open Source implementations being more reliable but it seems all servers have more or less the same level of reliability.

Usability

- OpenLDAP
  - Open standards so all operating systems/applications can use it as directory server.
  -
- FDS
  - Open standards so all operating systems/applications can use it use it as directory server.
- AD
  - Support for other operating systems such as UNIX is available, so other systems/applications can use it as directory server.
  - Great integration with other Microsoft products.

Efficiency

- It is hard to make a comparison for the directory servers regarding efficiency. The hardware resources necessary for the directory servers is more or less the same, but very dependent on the scale and functionality of your network.

Maintainability

- OpenLDAP
  - Less documentation
  - Pay support available from support companies.
  - Console based maintenance is not so easy.
- FDS
  - Good documentation
  - Pay support available from support companies or the commercial Redhat version.
  - Easy to maintain through graphical user interface and web interface.
- AD
  - Lots of documentation available.
  - Pay support available from support companies, many trained professionals available.
  - Easy to maintain through graphical user interface.

Portability

- OpenLDAP
  - Runs on multiple operating systems for instance Windows and UNIX.
  - Can be converted to another directory server. That makes it easier to change your server software if needed, for example when two companies start working together.
  - Hard to install because all has to be done by command line
- FDS
  - Runs on multiple mainly UNIX based operating systems.
  - Can be converted to another directory server. That makes it easier to change your server software if needed, for example when two companies start working together.
  - Easier to install because there is more assistance.
- AD
  - Runs only on Microsoft Windows Server.
  - Can be converted to another directory server except for Windows only features (if you use them). That can make it a bit harder to change the server software.
  - Easy to install because everything is done with a graphical user interface.

Another issue that can be important is that Active Directory requires a license fee; OpenLDAP and Fedora Directory Server are free to use.

The security aspect is very important as well, but hard to quantify. The directory servers are all quite mature and improved over the years. Security should be quite ok, but to measure the level of security and to compare the directory servers against each other quite some research needs to be done. The documentation that is available about security is mostly written by the companies themselves or by so called fan boys. An independent research should be done to compare the directory servers. That proves difficult as it costs quite some time and the directory servers are still improved. So to keep the results up to date new research should be done regularly. Unfortunately that is not done, but one can assume that they are quite secure since they are often used and serious bugs should be fixed within a short period of time.

It is possible to use both Active Directory and Fedora Directory Server in one network so that Windows computers can connect to Active Directory and UNIX computers can connect to Fedora Directory server. If that is a solution you would like to use in practice is something else as you have to maintain two different systems. Pay attention that if you would like to migrate from Active Directory to something else like OpenLDAP then you will not be able to use the Microsoft specific functionality. It is important to think about possible future changes for your directory server as many companies start working together or buy other companies. That results in directory servers that need to work together or should be combined into one directory server.

OpenLDAP is in practice not such a good solution, based on my personal experience it is hard to setup and maintain due to the lack of a user interface. Further documentation for OpenLDAP is a bit limited and there are no additional features such as organizational charts. Fedora Directory Server is a nicer solution, it is also free and it is possible to maintain the system with a graphical user interface and there is plenty of documentation.

So in the end Fedora Directory Server and Active Directory are the most serious alternatives we discussed in this thesis. There are other directory servers, but most

require a license or are not as advanced as Fedora Directory Server. In this case we wanted to compare a free version with a paid version, so we stick with Fedora Directory Server as one of the most advanced free versions and Active Directory as it is one of the most used directory servers.

Which of the two you should choose depends on the features you want, what network configuration you have and your budget. The following questions are important to ask when you have to choose between directory servers:

Do you need the extra Microsoft-only functionality Active Directory offers?
One of the most important reasons to choose Active Directory is if you want the extra functionality it offers for Microsoft operating systems. That functionality is not offered by other directory servers, so if you want it you can pick Active Directory, but you should remember that you will need to stay with Active Directory in order to use the extra functionality.

What is the budget?
If the budget is small and you cannot afford licenses than Fedora Directory server might be the best option. However maintenance costs should also be regarded, but it goes past the scope of this thesis to discuss which directory server is cheapest overall.

Which operating system is preferred by your administrators, or in which one are they most experienced?
If your administrators are experienced Microsoft or UNIX users then it might be easier to stay with that operating system when picking a directory server. That makes the setup and maintenance easier. However it is possible that the operating systems on the workstations are different or there is an indication that they will become different than the ones the administrators are most experienced with. Then you have to consider the support for those operating systems as well and it might be better to choose another directory server.

If the administrators are not really experienced then it is important to choose a directory server with good documentation or with (optional) paid support.

What computers do you have in your network?
If all workstations have the same brand of operating systems (all Microsoft or all Linux …) and the indication is that it will stay the same, then you can best choose the directory server according to that. So if you have a Microsoft based network then choose Active Directory, if it is UNIX based choose Fedora Directory Server.

If operating systems vary across workstations or if there is an indication that they might vary in the future then it is harder to choose a directory server. The Open Source ones are implemented to support multiple operating systems, but Active Directory has better support for Microsoft operating systems. Also because of the fact that Active Directory is pretty much the standard, most software builders add support for Active Directory. The organization might also opt for two different directory servers, one for Microsoft clients and one for other clients. Or maybe the organization has arguments to pick one single directory server based on budget, administrator experience or something else.

What directory server are your applications designed for?
If you have or plan to get applications that use a specific directory server than it might be an argument to pick that directory server instead of changing the application(s). In the end it is possible for 'Windows' applications to use Fedora Directory Server and for 'UNIX' applications to use Active Directory, but it remains a fact that some applications

have better support for some directory server. Applications written in .NET for instance can be implemented with Active Directory support quite easily and it is well documented. Those applications can also be implemented with Fedora Directory Server but that might take some more time and there is not so much documentation available.

If it is still unclear which directory server suits best for the organization than it might be best to use two directory servers. One for Microsoft operating systems and one for other operating systems as it might be easier for applications to use a 'pure' LDAP implementation. Active Directory does not completely comply with the LDAP standard and some non Microsoft platforms such as Java have mechanisms that work better with 'pure' LDAP implementations.

In most cases it does not matter too much which directory server you pick. Both Active Directory and Fedora Directory Server can give you the same (basic) functionality. Only if you need extra functionality for instance the Windows-only functionality from Active Directory then Active Directory should be picked. Otherwise it is best to depend on the administrators experience and the operating systems and applications in your network. Make sure everything keeps working and try to make it as maintainable as possible. If applications or operating systems require an extra directory server because they are build for that one, then add that directory server. The time it takes to align applications and systems with another directory server will possible be harder than adding an extra directory server and maintaining that one. In the future it is then possible to move everything on one server if software is changed. It is also possible to synchronize two directories for instance Active Directory and OpenLDAP. If it is not necessary to add an extra directory server then do not do that as it makes the maintenance a bit harder. Adding an extra server does not only make the maintenance harder, it is also harder from a security perspective as you have to maintain two directory servers which are probably used on two different operating systems. That makes it harder to keep those servers secure but it removes the single point of failure. The organization should see if they have a specific reason to use one or more directory servers (which might be the same directory server) and then decide what to do.

Almost every organization will have some arguments to pick a specific directory server (or multiple directory servers). If there is no reason at all Active Directory is probably the best pick. Active Directory is well documented and offers the same functionality as Fedora Directory and there are lots of companies that offer paid support. If you use Active Directory without the Windows-only functionality then it is also possible to transform you directory server to Fedora Directory Server or something else.

6 Java Authentication and Authorization Service

In the previous chapter directory servers where covered. For this thesis the directory servers are used to communicate with. This chapter and the following contain information about how to authenticate and authorize with the help of a directory server. That directory server contains the usernames and passwords for the users in the network.

In this chapter Java Authentication and Authorization service (JAAS) is described and some example code is given which I used to authenticate and authorize using a LDAP backend. First the relevant JAAS classes are covered so that the examples are easier to understand. After the coverage of the classes some examples that use LDAP can be found. The example code should not be seen as a guideline about how you should implement it, there might be better (more efficient) implementations. If you want to implement these things yourself, have a look at the tutorials from SUN. The examples were made to see how JAAS works and to figure out which advantages Java has compared to .NET regarding authentication and authorization with LDAP. In a later chapter that comparison is discussed in detail.

The JAAS is an integrated package since J2SDK 1.4, in previous versions it was an optional package. JAAS can be seen as a security framework for Java which defines how the authentication and authorization should be implemented. JAAS is maintained and developed by SUN and they provide some documentation online[14]. This chapter starts with an introduction to JAAS and the classes of JAAS are described and explained. If you are already familiar with JAAS then it is possible to skip the following paragraph and have a look at the examples I made. These examples are made to get an idea about how JAAS works compared to .NET. In the following chapter the authentication and authorization in .NET is explained. In chapter 9 the comparison between JAAS and .NET can be found.

6.1 JAAS overview

In this paragraph the different classes of JAAS are explained by theory and examples. If you are unfamiliar with JAAS then it is a good idea to first read this paragraph before looking at the examples. This paragraph serves as an introduction to the examples so that it is easier to understand how they work.

6.1.1 Authentication and authorization classes

There are some general classes that are used for both authentication and authorization: *Subject*, *Principals* and *Credentials*. These are described first and in the following paragraphs specific classes for authentication and authorization are described.

The *Subject* is an entity for example a user or service. In JAAS *Subject* is used to represent the source of a request. For authentication and authorization the *Subject* needs to be authenticated and authorized. After authentication the *Subject* is bound to one or more identities or *Principals*. *Principals* can be many things such as the name or unique number a user has. The *Subject* can also have *Credentials* which are security attributes of a user. There are two kinds of credentials: public and private. Both are stored in separate credential sets.

[14] See also Sun's JAAS Reference Guide:
http://java.sun.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html

If you write an application it is not necessary to instantiate a *Subject*. If a *LoginContext* is constructed without a Subject then an empty *Subject* is instantiated.

To perform an action as a particular *Subject* there are some methods available: *doAs(final Subject subject, final java.security.PriviligedAction action)* and *doAsPrivileged(final Subject subject, final java.security.PrivilegedAction action, final java.security.AccessControlContext acc)*. The difference between the two is that *doAs* uses the current threads *AccessControlContext* while the *doAsPrivileged* method can use another *AccessControlContext*.

This can be explained using an example from Sun's JAAS Reference Guide (29). This example assumes that a user named "Bob" has been authenticated (using *LoginContext* which will be described later). The *Subject* is populated with a *Principal* of class com.ibm.security.Principal and the name "Bob".

The policy file to grand Bob permission to read "foo.txt":

```
grant Principal com.ibm.security.Principal "BOB" {
    permission java.io.FilePermission "foo.txt", "read";
};
```

The action code:

```
class ExampleAction implements java.security.PrivilegedAction {
    public Object run() {
        java.io.File f = new java.io.File("foo.txt");

        // the following call invokes a security check
        if (f.exists()) {
            System.out.println("File foo.txt exists");
        }
        return null;
    }
}
```

The sample application code assuming the piece of code for the authentication process is executed before:

```
public class Example1 {
    public static void main(String[] args) {

        Subject bob;
        // Set bob to the Subject created during the
        // authentication process

        // perform "ExampleAction" as "BOB"
        Subject.doAs(bob, new ExampleAction());
    }
}
```

6.1.2 Authentication classes

There are some specific classes used for authentication. For the authentication process in JAAS the following steps are executed when authenticating a *Subject*:

- *LoginContext* is instantiated
- The configuration file is used by *LoginContext* to load the *LoginModules*
- The login method from *LoginContext* is invoked
- All *LoginModules* are invoked by the login method. Every *LoginModule* tries to authenticate the *Subject*. If authentication is successful then the *LoginModule* couples *Principals* and *Credentials* with the *Subject* object.
- The authentication status is returned by *LoginContext*.

- The Subject is provided by *LoginContext* when authentication is successful.

There are three classes used for authentication: *LoginContext*, *LoginModule*, *CallbackHandler and Callback*.

The *LoginContext* is used to authenticate subjects therefore he uses the *LoginModules*. *LoginModules* are 'plug and play' you can change them without modifying the application itself.

The *LoginModule* is an interface which can be used by developers to develop their own implementations of the *LoginModule*. It is not necessary to develop your own *LoginModule*, there are already some implementations available, but if you want a new *LoginModule* then you can develop one. Some common login modules present in the Java API or by other vendors:

- JndLoginModule
- Krb5LoginModule
- NTLoginModule
- UnixLoginModule
- NTActiveLoginModule (IBM)

The *CallbackHandler* is an interface which can be used to get authentication information from the user. The interface is then passed to the *LoginContext* which passes it to the *LoginModules*. The *LoginModule* can use the *CallbackHandler* to get or send information to the user. *LoginModules* can send an array of *Callbacks* (which is also an interface) to the *CallbackHandler*. The *CallbackHandler* can be implemented separately from the *LoginModule* it is easier to use implement various ways of user communication with the same *LoginModule*.

6.1.3 Authorization classes

The JAAS authorization can be used to grant access based on who is running the code and what code is being used.

The following steps are executed when authorizing a *Subject*:

- User is authenticated
- Access control context should be associated with the authenticated *Subject*
- The security policy should contain some entries

There are three classes used for authorization: *Policy*, *AuthPermission* and *PrivateCredentialPermission*.

Policy is an abstract class used to describe the access control policy. For JAAS you can use a file-based subclass which looks more like a configuration file to enter your policy. The *Authpermission* object is used to guard access to Subject, LoginContext, Policy and Configuration objects. *PrivateCredentialPermission* protects access to the *Subject*'s private credentials.

6.2 JAAS examples

This chapter contains some JAAS examples that are made for this thesis to be able to compare JAAS against .NET. The simple examples are for non web based applications, while the web applications example is for web applications. The applications need to connect to an LDAP server, mostly the data necessary to find that server is stored in some configuration file (like krb5.ini) on your computer. It is also possible to add something like the following commands when running the applications:

```
-Djava.security.krb5.realm=NWTRADERS.MSFT -Djava.security.krb5.kdc=192.168.1.200
```

The connection with LDAP is necessary so that the example applications can connect to the LDAP server and check whether the username and password are correct and see if that user is authorized to perform certain actions.

6.2.1 Authentication, simple JAAS example

This is a simple console based application that authenticates a user with the help of an LDAP server. It hopefully helps to understand the basics so that the later examples are easier to follow.

First there is the configuration file where the specific login module is set. The login module can be changed without having to modify the rest of the code. The Krb5LoginModule in the example configuration file below is used to work with Kerberos which is the standard authentication protocol for Active Directory. When you want to use Windows credentials (username and password of logged in user) then you should set useTicketCache to true. Debugging information can be helpful when debugging, you can set debug to true to show debugging information.

jaas.conf

```
JaasExample {
   com.sun.security.auth.module.Krb5LoginModule required useTicketCache=false
debug=false;
};
```

The Windows credentials are not used so the callback handler will ask for the credentials and try to authenticate the user with LDAP.

The example application below tries to authenticate to the LDAP server with the username and password that you have to provide.

SimpleAuthProto.java

```
import javax.security.auth.login.LoginContext;
import javax.security.auth.login.LoginException;
import com.sun.security.auth.callback.TextCallbackHandler;

public class SimpleAuthProto {

    public static void main(String[] args) {

        LoginContext lc = null;

        try {
            lc = new LoginContext("SimpleAuthProto", new TextCallbackHandler());
            lc.login();
            System.out.println("Authentication succeed!");

            lc.logout();
        }
        catch (LoginException le) {
            System.err.println("LoginContext: Authentication failed");
            le.printStackTrace();
            System.exit(0);
        }
        catch (SecurityException se) {
            System.err.println("LoginContext: Security exception");
            se.printStackTrace();
            System.exit(0);
        }
```

```
    }
}
```

To compile and run the application the following commands are necessary:

```
javac SimpleAuthProto.java
java -Djava.security.auth.login.config==jaas.conf SimpleAuthProto
```

6.2.2 Authorization, JAAS with policy example

The previous example only used authentication, this one uses authorization as well. The program first authenticates a user and if the user is 'anna' then the user is authorized to read a file. The content of the file is displayed if the correct user is authenticated.

The configuration file is almost the same as the previous one.

JAASConfig.conf

```
JAASPolicyProto {
   com.sun.security.auth.module.Krb5LoginModule required useTicketCache=false
debug=false;
};
```

In the policy file all authorizations are listed. This application needs to authenticate therefore the application needs some permission, those are listed below. Further the permission for 'anna' to read the file is listed.

JAASPolicy.policy

```
grant codebase "file:./JAASPolicyProto.jar" {
    permission java.security.AllPermission "MonitoringAuth.*";
    permission javax.security.auth.AuthPermission "createLoginContext.JaasExample";
    permission javax.security.auth.AuthPermission "doAsPrivileged";
};

grant codebase "file:./JAASPolicyProtoAction.jar", Principal
javax.security.auth.kerberos.KerberosPrincipal "anna@NWTRADERS.MSFT" {
    permission java.io.FilePermission "test.txt", "read";
};
```

An implementation of PrivilegedAction used to try and read the file which invokes a check if the user is authorized.

JAASPolicyProtoAction.java

```
import java.io.FileReader;
import java.io.BufferedReader;
import java.security.PrivilegedAction;
import java.io.IOException;

public class JAASPolicyProtoAction implements PrivilegedAction {

    public Object run() {
        try{
            BufferedReader br = new BufferedReader(new FileReader("test.txt"));
            String s;
            while((s = br.readLine()) != null) {
                System.out.println(s);
            }
        }
        catch (IOException e) {
            System.out.println("IOException");
```

```
        }
        return null;
    }
}
```

Below you see the content of the file which is readable for the user 'anna'.

Test.txt

```
You have got authorization to read this file.
```

The test application authenticates the user and then tries to execute the PrivilegedAction as the authenticated user.

JAASPolicyProto.java

```
import javax.security.auth.login.LoginContext;
import com.sun.security.auth.callback.TextCallbackHandler;
import javax.security.auth.Subject;
import javax.security.auth.login.LoginException;
import java.security.PrivilegedAction;

public class JAASPolicyProto {

    public static void main(String[] args) {

        LoginContext lc = null;

        try {
            lc = new LoginContext("JAASPolicyProto", new TextCallbackHandler());

            lc.login();
            System.out.println("Authentication succeed!");
        }
        catch (LoginException le) {
            System.err.println("LoginContext: Authentication failed");
            le.printStackTrace();
            System.exit(0);
        }
        catch (SecurityException se) {
            System.err.println("LoginContext: Security exception");
            se.printStackTrace();
            System.exit(0);
        }

        Subject sub = lc.getSubject();
        PrivilegedAction act = new JAASPolicyProtoAction();
        Subject.doAsPrivileged(sub, act, null);
        lc.logout();
    }
}
```

To compile and run the application the following commands are necessary:

```
javac JAASPolicyProtoAction.java JAASPolicyProto.java
jar -cvf JAASPolicyProtoAction.jar JAASPolicyProtoAction.class
jar -cvf JAASPolicyProto.jar JAASPolicyProto.class
java -classpath JAASPolicyProto.jar;JAASPolicyProtoAction.jar -Djava.security.manager -
Djava.security.policy=JAASPolicy.policy -
Djava.security.auth.login.config=JAASConfig.conf JAASPolicyProto
```

6.2.3 Web application, JAAS with Tomcat example

Nowadays lots of applications are web applications, to see how JAAS works with web applications I build an example. This example uses a callback handler to get the username and password the user entered in the form on the webpage. If the user can be authenticated against LDAP then the user has authorization to read the file and the content of the file is displayed.

AuthenticateUser.java

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.security.auth.callback.*;
import javax.security.auth.login.*;
import javax.naming.*;
import javax.naming.ldap.*;

public class AuthenticateUser extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");

        String name = request.getParameter("name");
        String password = request.getParameter("password");

        if (name != null || password != null) {
            try{
                NamePasswordCallbackHandler npch = new
NamePasswordCallbackHandler(name, password);
                LoginContext lc = new LoginContext("AuthenticateUser", npch);
                lc.login();
                out.println("<i>Authentication was succesfull</i><br>");

                out.println("<i>You are authorized and can see the text of the file
below:</i><br>");
                BufferedReader br = new BufferedReader(new FileReader("C:/Program
Files/Apache Software Foundation/Tomcat 6.0/webapps/examples/WEB-
INF/classes/test.txt"));
                String s;
                while((s = br.readLine()) != null) {
                    out.println(s);
                }
                lc.logout();

            }
            catch (IOException e) {
                out.println("You have no authorization to read the file");
            }
            catch(Exception e){
                out.println("Probably authentication failed");
            }
        }
        out.print("<form action=\"");
        out.print("AuthenticateUser\" ");
        out.println("method=POST>");
        out.println("Name: ");
        out.println("<input type=text size=10 name=name>");
```

```
          out.println("<br>");
          out.println("<br>");
          out.println("Password: ");
          out.println("<input type=text size=10 name=password>");
          out.println("<br>");
          out.println("<input type=submit>");
          out.println("</form>");
          out.println("</body>");
          out.println("</html>");
     }

     public void doPost(HttpServletRequest request,HttpServletResponse response) throws
IOException, ServletException {
          doGet(request, response);
     }

     public class NamePasswordCallbackHandler implements CallbackHandler {
          private String name;
          char[] password;

          public NamePasswordCallbackHandler(String name, String password) {
               this.name = name;
               this.password = password.toCharArray();
          }

          public void handle(Callback[] callbacks) throws java.io.IOException,
UnsupportedCallbackException {
               for (int i = 0; i < callbacks.length; i++) {
                    if (callbacks[i] instanceof NameCallback) {
                         ((NameCallback)callbacks[i]).setName(name);
                    } else if (callbacks[i] instanceof PasswordCallback) {
                         ((PasswordCallback)callbacks[i]).setPassword(password);
                    } else {
                         throw new UnsupportedCallbackException(
                                   callbacks[i], "Callback class not supported");
                    }
               }
          }
     }
}
```

Below you see the content of the file which is displayed if the user is authenticated.

Test.txt

```
You have got authorization to read this file.
```

Tomcat requires some configuration to be able to use the web application in the web server. Below you can see the necessary rules for the sample application.

web.xml

```
    <servlet>
        <servlet-name>AuthenticateUser</servlet-name>
        <servlet-class>AuthenticateUser</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>AuthenticateUser</servlet-name>
        <url-pattern>/servlets/servlet/AuthenticateUser</url-pattern>
    </servlet-mapping>
```

6.2.4 Java Naming and Directory Interface (JNDI) example

There are more ways in Java to achieve authentication as alternative to JAAS, for instance JNDI. Although JNDI does not offer as much options as JAAS it might be an alternative for simple authentication jobs. JAAS however remains the standard in JAVA for all authentication and authorization tasks. Below you can find a small example to illustrate JNDI. The application tries to authenticate the user 'anna'.

JNDIProto.java

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.directory.DirContext;
import javax.naming.ldap.InitialLdapContext;
import javax.naming.NamingException;

public class JNDIProto
{
    public static void main (String[] args)  {

        String LDAPURL = "ldap://192.168.1.200:389/DC=server01,DC=nwtraders,DC=msft";
        Hashtable env = new Hashtable();

        env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
        env.put(Context.PROVIDER_URL, LDAPURL);
        env.put(Context.SECURITY_AUTHENTICATION, "simple");
        env.put(Context.SECURITY_PRINCIPAL,"anna");
        env.put(Context.SECURITY_CREDENTIALS, "p@ssw0rd");

        try {
            DirContext con = new InitialLdapContext(env,null);
            // Authentication succeed
            System.out.println("Authentication succeeded");
            con.close();
        }
        catch(NamingException ne) {
            // Authentication failed
            System.err.println("Authentication failed");
            ne.printStackTrace();
        }
    }
}
```

Commands to compile and run the application:

```
javac JNDIProto.java
java JNDIProto
```

6.3 Conclusion

To make an application with authentication and authorization with JAAS is not a big problem. Online SUN provides quite some documentation which should be enough to build applications. The problem however is that when you encounter a problem then it is hard to find an answer. When I was making the examples and wanted to set the authorizations in the policy file I encountered the problem that I had to set all the permissions for authentication in order for the application to work. It was not possible to set some sub permissions and on the internet I could not find documentation about the problem. It might be a problem with the Java version I used as some examples on the internet which contained the same kind of policy gave the same issue. However I was not able to find a solution which might be a problem for people actually using JAAS. If there are no (simple) solutions then people often build less secure applications that have more rights than necessary.

7 Microsoft .NET

In the previous chapter some Java examples where explained, this chapter will introduce some .NET examples. These .NET applications will be used as a comparison to the Java examples. Note again that they are just simple examples which are not perfect at all. If you want to implement something similar have a look at the' MSDN Securing ASP.NET Web Sites'[15] site. There web application security is explained including the membership provider and role provider. In .NET those providers are used for authentication and authorization. In the following paragraph some more information about those providers is given. The examples in this chapter are not based on membership or role providers, but with the information in the following paragraphs it should be easy to implement those providers.

This chapter is structured in the same way as the previous one about JAAS. The chapter starts with an introduction about authentication and authorization in. If you are already familiar with authentication and authorization in .NET then you might opt to skip the first paragraph. The second paragraph includes the example .NET applications I made and after that a technical comparison of JAAS and .NET is made. In that technical comparison the components of JAAS and .NET are compared to show what functionality is built in which components of JAAS and .NET.

7.1 .NET overview

In this paragraph an overview of the authentication and authorization in .NET is given. The paragraph serves as an introduction for the following paragraph which includes some sample applications. In this paragraph the different kinds of authentication in .NET are explained, each one has a specific purpose and they are used in different scenarios. After that introduction providers are covered, the providers provide an abstraction layer for authentication and authorization.

7.1.1 Authentication

ASP.NET gives some standard authentication modes that you can use to prove the identity of a user:

- Forms authentication
- Windows authentication
- Passport authentication

The different modes of authentication are used in different scenarios. Forms authentication is used when building web applications. Windows authentication is mostly used when building applications that are not web based. Passport authentication is used by for instance MSN and the original idea was to use it as a single sign on mechanism. So which authentication mode you need depends on which sort of application should be build.

How each authentication mode works is discussed in the following pages.

The authentication mode is set in the web.config file as follows:

```
<configuration>
    <system.web>
        <authentication mode="[Windows/Forms/Passport/None]">
        </authentication>
    </system.web>
```

[15] http://msdn2.microsoft.com/en-us/library/91f66yxt.aspx

```
</configuration>
```

Forms authentication
If you have a website you probably want forms based authentication. Forms authentication uses cookies to store the user name and password. Those credentials are stored in a database or text file and are used during the session. If a user is not logged in and requests for a secure page then the user is directed to the login page.

The following steps are performed when someone requests a page:

- ASP.NET checks if there is a session cookie.
    - If so ASP.NET assumes that the person is already authenticated and the request is processed.
    - If not the person is redirected to a login form. There the person can authenticate and the session cookie is stored.

To use form based authentication you have to add the following to the web.config file:

```
<configuration>
    <system.web>
        <authentication mode="Forms">
        </authentication>
        <forms name="login" loginUrl="login.aspx" />
        <authorization>
            <deny users="?" />
        </authorization>
    </system.web>
</configuration>
```

Where the statement <deny users="?"> grants permission to all authenticated users and denies all users that are not authenticated. The "?" stands for all users that are not authenticated or are anonymous users. If you want to allow all users than you have to use <allow users="*">. Make sure that you put the elements in the correct order since the first element has higher priority than the lower ones.

You can also add the credentials in the web.config file under the forms tag. Then you do not need to request them from the user. Therefore you have to add the following:

```
<credentials passwordFormat="Clear">
    <user name="testUser" password="testPassword" />
</credentials>
```

If you just want to authenticate a user then the C# code for forms authentication will look something like this:

```
if (checkUser (userName.txt, userPassword.txt))
{
    FormsAuthentication.RedirectFromLoginPage(userName.txt, False);
}
else
    lblInfo.Text = "The username or password supplied are incorrect";
}

private checkUser(string userName, string userPassword)
{
    //Connect to the database and verify the credentials
}
```

In this example checkUser is used to check the supplied credentials. If those credentials are correct then the user is redirected. The static method

RedirectFromLoginPage is used to create an authentication ticket for the rest of the session.

Windows authentication
The default authentication mode of ASP.NET uses the users Windows account credentials to authenticate. That is done by IIS which performs the authentication and sends the authenticated identity to the code. This can be used to provide SSO, a user just uses his username and password to login to Windows and then the applications can use that username and password for authentication and authorization.

There are four types of windows authentication:

- Anonymous authentication: IIS allows every user to access the application.
- Basic authentication: the Windows credentials are used, but are being sent in plain text over the network so it is not secure.
- Digest authentication: the same as basic, except with this the password is hashed. You need Internet Explorer 5.0 or above.
- Integrated Windows authentication: the password is not sent over the network. The application makes use of Kerberos or challenge response protocols to authenticate.

Passport authentication
This authentication mode uses Microsoft Passport service and can be used to create a single sign in service for all sites that implemented the passport single sign in service. To configure your website to use passport authentication you have to add something like the following:

```
<configuration>
    <system.web>
        <authentication mode="Passport">
            <passportredirectUrl="login.aspx" />
        </authentication>
        <authorization>
            <deny users="?" />
        </authorization>
    </system.web>
</configuration>
```

Custom authentication
With ASP.NET you can also implement your custom authentication then you should specify the authentication mode as "none" in the web.config file.

7.1.2 Authorization

As with authentication .NET provides some standard implementations for authorization. The three standard types of authorization in .NET are:

- URL authorization: in web.config you can specify authorization rules for files and directories.
- File authorization: uses the NTFS file permission for granting access.
- Role checks: use the user's role membership for instance for RBAC.

It is possible that you want a process to act as the authenticated or another specified user. This technique is known as impersonation and .NET can take the identity that is provided by IIS (Internet information Services). Impersonation using Microsoft IIS to authenticate the user and pass an authenticated token to the ASP.NET application.

NTFS can then be used to regulate access since the ASP.NET application relies on the NTFS settings.

Impersonation is disabled by default and can be set using the web.config file with optionally supplying an identity that should be used for all resource requests:

```
<identity impersonate="[true/false]" username="testUser" password="testPassword" />
```

The following steps are done in the authentication and authorization process when a new request arrives:

- IIS checks validity of the request.
    - If the authentication mode is anonymous (default): request is authenticated.
    - If the authentication mode is not anonymous: IIS performs the specified authentication mode before passing the request to ASP.NET.
- ASP.NET checks whether impersonation is enabled or not
    - If impersonation is enabled: the application executes using the identity of the entity on behalf of which it is executing the task.
    - If impersonation is disabled: the application executes using the identity of the IIS local machine's identity and the privileges of the ASP.NET user account.
- The identity that is authenticated is used to request resources from the operating system.
- ASP.NET performs a check on the requested resources to see if the user is authorized to use them. ASP.NET returns the request through IIS.

The following figure shows how the authentication and authorization process is executed.

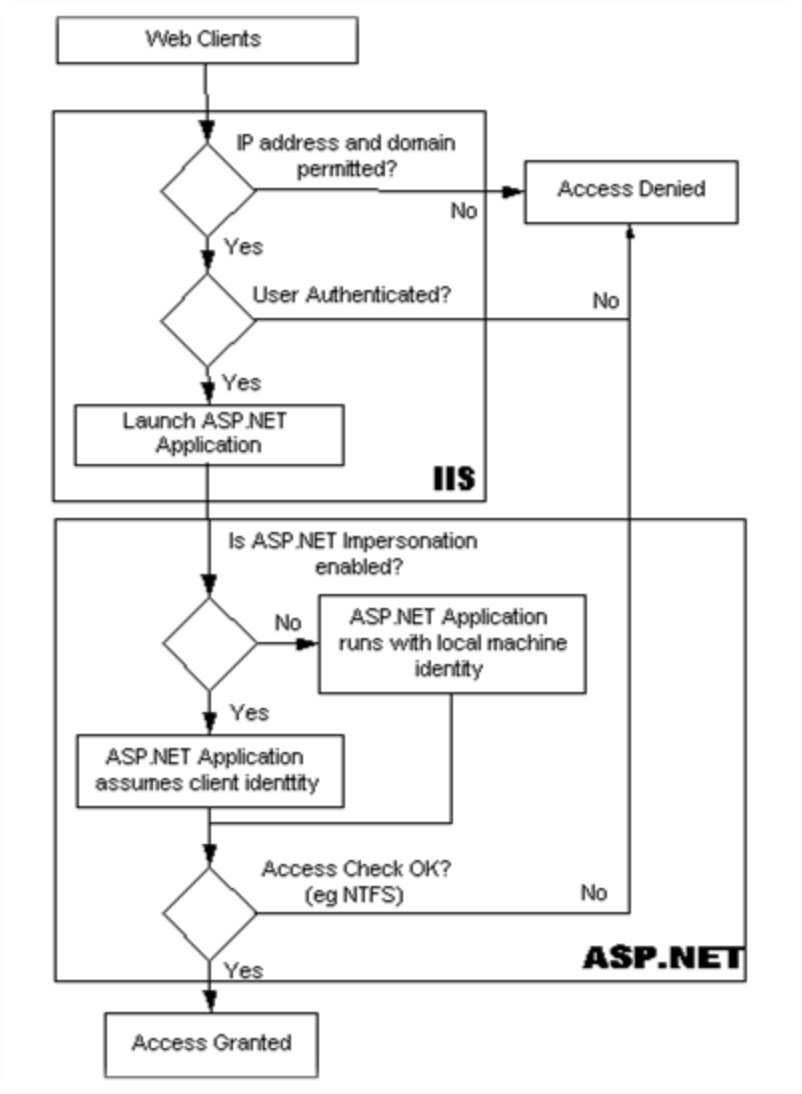


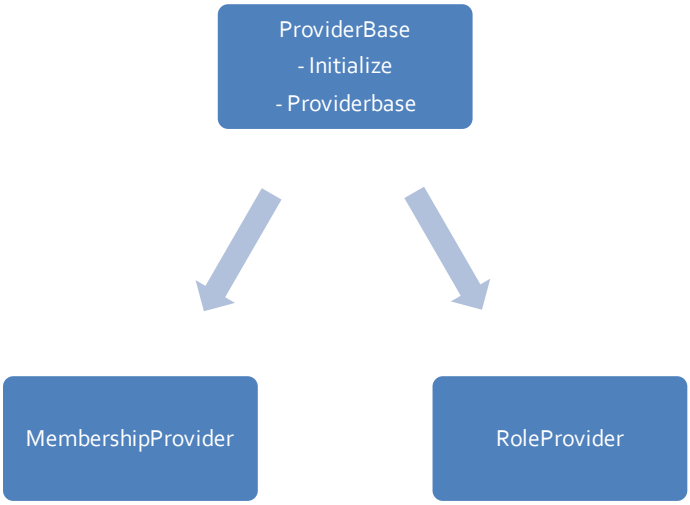
Figure 13 Authentication process[16]

7.1.3 Authentication and authorization with providers

With ASP.NET 2.0 there are some implementations that make programming a bit easier. Before ASP.NET 2.0 you had to write code to accept and verify user's credentials and to manage the users. Mostly the code to achieve this is more or less the same. ASP.NET 2.0 provides membership and role providers to manage roles and memberships in web applications. The providers can be used to provide authentication and authorization services to the applications you build.

The provider model is based on the abstract class ProviderBase which has two methods: initialize and providerbase. MembershipProvider and RoleProvider inherit from ProviderBase and are also abstract classes. Their relation is shown below in the following picture.

[16] http://msdn2.microsoft.com/en-us/library/ms978378.authaspdotnet01(en-us,MSDN.10).gif



The MembershipProvider and RoleProvider add extra properties and methods and define the interface for their specific functionality. ASP.NET provides a SqlMembershipProvider and an ActiveDirectoryMembershipProvider, additional MembershippProviders can be implemented. If you want to use XML files instead of SQL server or Active Directory then you can build it yourself or look for available MembershipProviders on the internet. The advantage is that you do not have to care about the things that happen behind the interface. Whatever you use as a database the methods to access the data remain the same. The methods of RoleProvider and MembershipProvider are displayed below to give an idea of the functionality they provide.

RoleProvider	**MembershipProvider**
AddUsersToRoles	*ChangePassword*
CreateRole	*ChangePasswordQuestionAndAnswer*
FindUsersInRole	*CreateUser*
GetAllRoles	*DecryptPassword*
GetRolesForUser	*DeleteUser*
GetUsersInRole	*EncryptPassword*
IsUserInRole	*FindUsersByEmail*
RemoveUsersFromRoles	*FindUsersByName*
RoleExists	*GetAllUsers*
RoleProvider	*GetNumberOfUsersOnline*
	GetPassword
	GetUser(+1 overload)
	GetUserNameByEmail
	membershipProvider
	OnValidatingPassword
	ResetPassword
	UnlockUser
	UpdateUser
	ValidateUser

Login controls
It is possible to use login controls to create a web form by just drag and drop without writing code. The login controls communicate with the membership provider. Controls can also be used for user maintenance; for instance for changing passwords.

Membership provider

A new feature of ASP.NET 2.0 is the membership which can be used together with forms authentication. The membership feature includes an API which you can use to validate users and maintain the user database. It provides a layer of abstraction for the underlying data in databases or Active Directory. Two common membership providers are the ActiveDirectoryMembershipProvider for Active Directory and the SQLMembershipProvider for SQL Server, custom providers can be made.

Mostly you would like to use login controls or some interface to communicate with the user. Maybe you do not want such things and then it is also possible to approach the MembershipProvider hardcoded:

```
MembershipCreateStatus status;

Membership.CreateUser(userName,userPassword,userEmail,userQuestion,userAnswer,userIsAppr
oved,out status);

if(status == MembershipCreateStatus.Succes)
{
    //perform the actions that you want
}
```

To validate a user with the membership the following code is used:

```
bool isValidUser = Membership.ValidateUser("userName@domain","p@ssw0rd");
```

The 'How to: Use Forms Authentication with Active Directory in ASP.NET 2.0'[17] on MSDN gives a tutorial for the active directory membership provider.

Role provider
You can define roles for instance for administrators, authenticated users and anonymous users. Then it is possible to grant or restrict access to these roles, which can be used to implement RBAC. That is easier then assigning rights to individual users. Roles are configured in the <authorization> section of web.config. These authorization settings can be made with the Web Site Administration Tool.

The RoleProvider class provides all functionality that you need for authorization in your application. As with the MembershipProvider you can define your own RoleProvider if you have additional wishes.

Every RoleProvider uses his own data source to store and retrieve information about the roles. In ASP.NET you already have the SqlRoleProvider which uses the SQL Server database, the WindowsTokenRoleProvider which uses the user's role information from the Windows group membership and the AuthorizationStoreRoleProvider which works with Microsoft Authorization Manager (AzMan). AzMan can use Active Directory as well as XML files to store role information.

The role manager is disabled by default, to enable it you have to add the following code to the web.config:

```
<system.web>
    <roleManager enabled="true" />
</system.web>
```

The following example shows how you can create roles, add a user to that role and check if the authenticated user is in the role:

```
if (!Roles.RoleExists("TestRole"))
{
```

[17] http://msdn2.microsoft.com/en-us/library/ms998360.aspx

```
    Roles.CreateRole(“TestRole”);
}

Roles.AddUserToRole(“userName”,”TestRole”);

if(Roles.IsUserInRole(“TestRole”))
{
    // Perform actions
}
```

7.2 .NET examples

In this paragraph the examples I made for .NET are covered. These examples are used in chapter 9 to compare Java and .NET. In this chapter there are some basic examples that do not used the providers that are covered in the previous paragraph.

7.2.1 Authorization, simple .NET example

This is a simple example with hardcoded username and password and it tries to authenticate to an LDAP server. If a user is authenticated then the user is authorized to read the test.txt file. The content of the file is displayed if the user is authenticated.

Authentication.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.DirectoryServices;
using System.IO;

class Authentication
{
    private static void Authenticate(string LDAPDomain, string userName, string
userPassword)
    {
        try
        {
            DirectoryEntry entry = new DirectoryEntry(LDAPDomain, userName,
userPassword);
            object nativeObject = entry.NativeObject;
            Console.WriteLine("User is authenticated");
        }
        catch (Exception e)
        {
            Console.WriteLine("User is not authenticate: " + e.Message);
        }

        StreamReader text = new StreamReader("../../test.txt");
        Console.WriteLine(text.ReadLine());
        text.Close();
    }

    static void Main(string[] args)
    {
        String ldapDomain = "LDAP://192.168.1.200";
        String userName = "CN=anna,CN=Users,DC=nwtraders,DC=msft";
        String userPassword = "p@ssw0rd";
        Authenticate(ldapDomain, userName, userPassword);
        Console.ReadLine();
    }
}
```

Test.txt

```
You have got authorization to read this file.
```

7.2.2 Web application, .NET example

This is a simple form based example where the user has to provide his username and password. If the username and password are present in the LDAP server then the user is authenticated and has authorization to read the test.txt file. The content of the test.txt file is displayed if the user is authorized.

LDAP.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="LDAP.aspx.cs" Inherits="LDAP" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>LDAP</title>
</head>
<body>
    <form id="form1" runat="server">
        <table>
            <tr>
                <td>User:</td>
                <td><asp:TextBox ID="name" runat="server"
Width="100px"></asp:TextBox></td>
            </tr>
            <tr>
                <td>Password:</td>
                <td><asp:TextBox ID="password" runat="server"
Width="100px"></asp:TextBox></td>
            </tr>
        </table>
        <asp:Button ID="btnLogin" runat="server" Text="Login" OnClick="btnLogin_Click"
/>
        <asp:Label ID="lblAuth" runat="server"></asp:Label>
        <br />
        <asp:Label ID="lblReadFile" runat="server"></asp:Label>
    </form>
</body>
</html>
```

LDAP.aspx.cs

```
using System;
using System.DirectoryServices;
using System.IO;

public partial class LDAP : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void btnLogin_Click(object sender, EventArgs e)
    {
        lblReadFile.Text = "";
        String ldapDomain = "LDAP://192.168.1.200";
        String userName = "CN=" + name.Text + ",CN=Users,DC=nwtraders,DC=msft";

        if (Authenticate(ldapDomain, userName, password.Text))
```

```
        {
            StreamReader text = new StreamReader(Server.MapPath("test.txt"));
            lblReadFile.Text = text.ReadLine();
        }
    }

    private bool Authenticate(string LDAPDomain, string userName, string userPassword)
    {
        try
        {
            DirectoryEntry entry = new DirectoryEntry(LDAPDomain, userName,
userPassword);
            object nativeObject = entry.NativeObject;
            lblAuth.Text = "User is authenticated";
            return true;
        }
        catch (Exception e)
        {
            lblAuth.Text = "User is not authenticated";
            return false;
        }
    }
}
```

Test.txt

```
You have got authorization to read this file.
```

7.3 Technical comparison

To get an overview of Java and .NET this comparison is included. It gives a short overview about which components in Java and .NET are responsible for which elements of identity management. This might give some support for people who are unfamiliar with (one of) the languages.

The authentication is handled by the following components:

- Java: loginModule
- .NET: authentication modes and Membership provider

The authorization is handled by the following components:

- Java: policy file
- .NET: Role Provider

These are the main elements that are responsible for authentication and authorization in the different languages.

7.4 Conclusion

Making the examples in .NET is not a big problem; there is lots of documentation available. The integration between Visual studio and Internet Information Services (IIS) is also very helpful to test the web applications. In my experience .NET is slightly easier then Java. The excellent documentation is really helpful where Java sometimes lacks documentation. Not only is the documentation helpful when building applications but also when errors occur and they need to be fixed. Another advantage of .NET is the excellent integration with the server (IIS) and other Microsoft products like Active Directory. The abstraction that is build into .NET in the last years is also very helpful to abstract from underlying databases for instance. A side note to this is that the integration and abstraction are (mainly) aimed at Microsoft products. So if you have

non-Microsoft software it is a bit harder to use with .NET. An example is the abstraction with the Membership provider which by default only supports Active Directory and SQL server. If you want to use non-Microsoft products then it is most likely that you have to add quite some extra code to achieve the same level of abstraction. Java offers a bit more support for non-Microsoft (and non-SUN) products; they support the most common used software.

8 Advanced forms of identity management

The previous two chapters include some sample applications. Those are quite simple identity management systems, but in the real world they are much more advanced. Things like self service, auditing options and so forth are implemented in identity management systems. In this chapter two elements of more advanced identity management are introduced. These forms can be found in the later stages of the identity management maturity models.

8.1 Service Oriented Architecture

Service Oriented Architecture or SOA is quite a buzzword the last few years. Many organizations want to redesign their architecture to SOA. There are quite some definitions of SOA available and on the internet there are quite some debates about what SOA is exactly. What follows is a definition of SOA which will be explained shortly, more (technical) information about SOA is widely available on for instance the internet.

Bieberstein, Bose, Fiammante, Jones and Shawn (30) define SOA as follows:
A service-oriented architecture is a framework for integrating business processes and supporting IT infrastructure as secure, standardized components (services) that can be reused and combined to address changing business priorities.

The idea behind this is that you make a mapping from a business process on a number of services that are necessary for that process. The financial administration can be seen as an example. For the administration the employees need some services to do their work. Those services can be seen as applications or just separated functionality from applications. All those services are coupled to what is called the Enterprise Service Bus (ESB). It is possible to build these services in Java, .NET or any other language and put them all on one ESB. Not only small services but also bigger packages like SAP can be attached to the ESB. The idea is that every business processes can use the services it needs through the ESB. The main benefit is that one can reuse services and does not have to build services multiple times for every application. For the business it is also profitable as the business and the technology are better aligned.

SOA has evolved over the years and both Sun and Microsoft support it and have made some frameworks and tools for it. What follows is a short introduction to SOA with Sun and with Microsoft.

Sun
It is possible to build a SOA from scratch with Java. On the internet some examples can be found where people have built it themselves with the help of some standards and frameworks. An example is the paper 'Identity as a Service' (31) where a SOA is described which works with Web services. They made a blueprint and tested their solution. It uses Enterprise Java Beans (EJB) and OpenLDAP. This shows that it is possible to build it yourself, but it is not easy. As chapter 10 shows there are quite some guidelines for identity management beside those guidelines there are also guidelines for SOA. So there are even more elements to keep in mind when designing identity management systems like these.

Sun also offers the Java Composite Application Platform Suite (Java CAPS). This package can be used to build a software infrastructure using the SOA approach. The package is based on standards and is an open and extensible platform. It was hard to

find some useful opinions about the product on the internet. Most information available is just advertisement or not really informative. That makes it hard to form an opinion on CAPS. If CAPS works then it might be a very useful alternative to building everything from scratch. A foundation would ensure that most features already work, they are based on standards and hopefully they comply with guidelines. Then the only thing left to do is extend and customize it to the desired level. That could save quite some work. So the idea is nice, the question however remains whether it is useful in practice.

Microsoft
Microsoft offers two main elements to support SOA: a framework and BizTalk a server to let applications communicate with each other.

The Windows Communication Framework (WCF) is a framework which can be used to build applications that need to communicate between each other. WCF is part of the .NET framework for communications and is introduced in the .NET 3.0 frameworks.

WCF can make use of Biztalk as a communication mechanism. In Figure 14 below is shown how a client can access the services through BizTalk. BizTalk can be seen as a sort of ESB which lets the client communicate with the services.

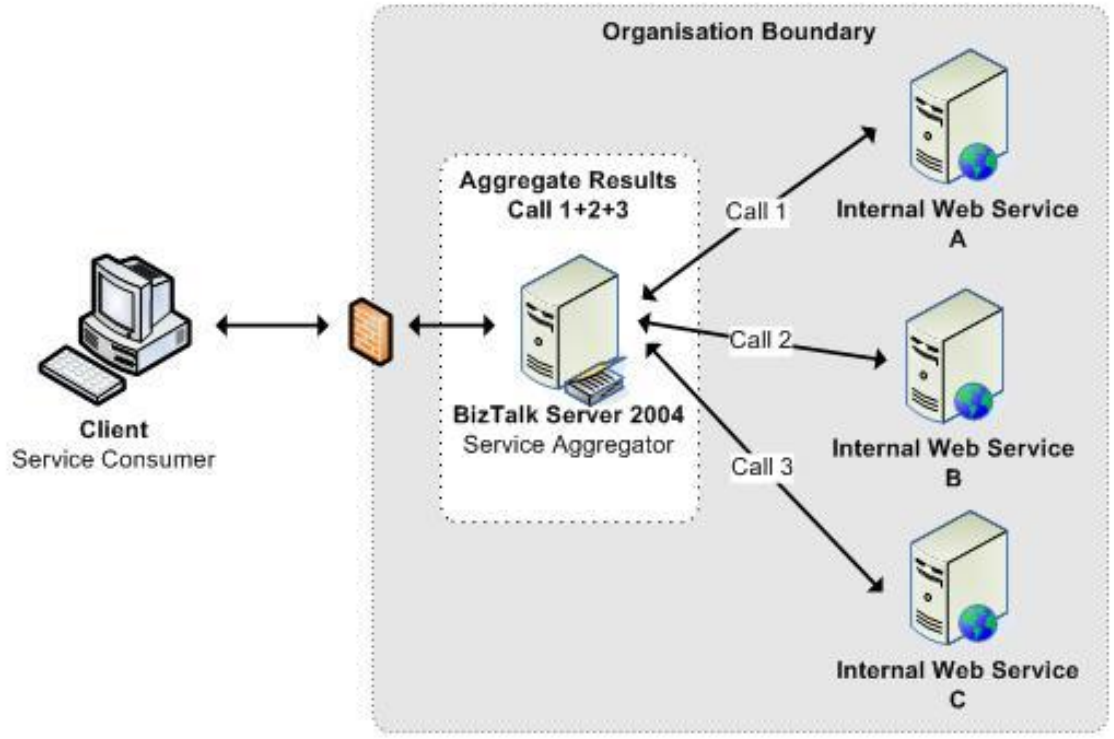


Figure 14 Service aggregator[18]

That guidelines are necessary to implement SOA in a quality way is shown by the available guidelines. One example is the 'WCF Security Guidance Project: Microsoft patterns & practices'[19]

[18] Source: http://www.topxml.com/rbnews/BizTalk-EAI/re-22433_BizTalk-and-SOA.aspx
[19] Source: http://www.codeplex.com/WCFSecurity

8.2 Federated identities

The concept of federated identities was explained before in paragraph 2.9. According to Wikipedia it has two meaning[20]:

- 'The virtual reunion, or assembled identity, of a person's user information (or principal), stored across multiple distinct identity management systems. Data is joined together by use of the common token, usually the user name.'
- 'The process of a user's authentication across multiple IT systems or even organizations.'

An example is the use of mobile phones where you often access the network of another provider then your own provider. That other provider still gives you access because it trusts your own provider. With federated identities the so called service provider forwards the authentication to the identity provider. The identity provider then lets the service provider know whether the user is allowed access or not.

Next the products of Sun and Microsoft are described that support federated identities.

Sun
With Java mainly JAAS is used with the Security Assertion Markup Language (SAML). SAML is based on XML and is a standard for exchanging authentication and authorization data between an identity provider and a service provider. SAML is also used to solve the SSO problem so that one needs just one identity.

In Figure 15 the process with service providers and identity providers is shown. This picture shows the use of SAML to authenticate Joe.

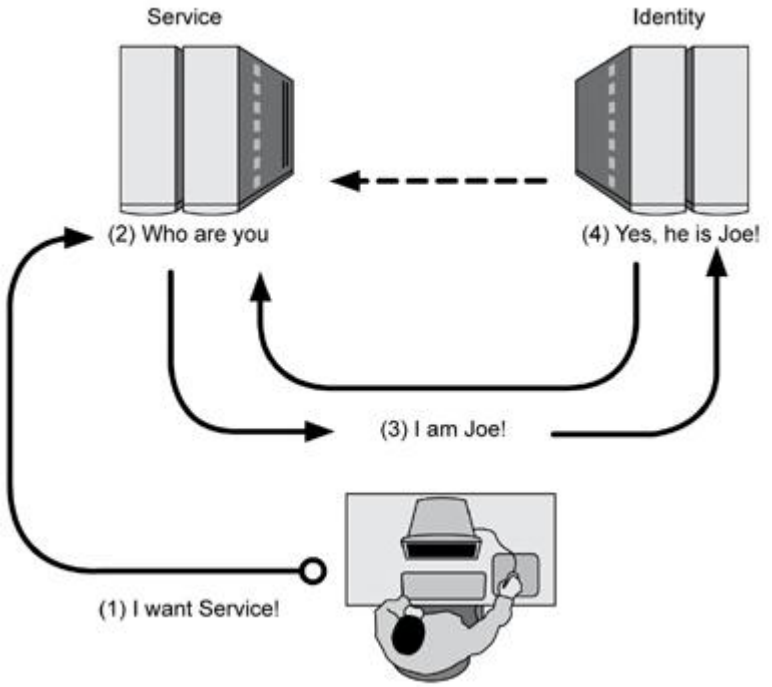


Figure 15 SAML

[20] Source: http://en.wikipedia.org/wiki/Federated_identity

As with SOA there are also packages available to support federated identities. Sun offers OpenSSO which is based on Sun's Java System Access Manager and Sun's Federation Manager. It is the public source variant of those two packages.

Microsoft
Microsoft offers the Active Directory Federation Services (ADFS) which are based on the WS-* standard. WS is a collection of communication protocols which are released by the Oasis-open organization. As with the previous products it is hard to say how helpful ADFS is as there was no really useful independent information available.

8.3 Conclusion

Hopefully this short chapter shows a bit more that it is quite hard to build an actual identity management system which has a high level of maturity. Not only does one have to use the standard identity management guidelines, but also extra guidelines for SOA and federated identities. As well as the many different frameworks and standards one should use to build these elements of identity management. The amount of code that has to be written and the elements one has to consider increases a lot when the identity management system becomes more complicated. For simple applications like the examples shown in this thesis it is not so hard, but the further one gets in the identity management maturity models the harder it is to implement the identity management system.

That complexity introduces the question whether it is easier and maybe better to use a standard package. Maybe a package that needs some customization and additions or a quite advanced package where everything is already implemented. What is important is that the time and money it costs to implement from scratch or to use or even buy a package are compared.

For future research it might be interesting to make some comparisons. The following comparisons would be quite interesting:

- Compare packages against applications build from scratch.
- Compare packages against packages that needs (some) customization.

The results from that research should give an insight into costs, necessary time and problems that are encountered afterwards. With the results of the research it should be easier for organizations to decide what to do. It is also interesting for organizations like Sun; for instance to see which areas of Java need improvement.

9 Comparison of Java and .NET regarding authentication and authorization with LDAP

Before beginning the research the expectation was that Java and .NET would have about the same level of functionality. It should be possible to do whatever you want to do with both platforms. But it was not clear if that would be the same in practice as there are not many comparisons made and the functionality of the platforms continuously improves, especially in .NET which had quite some releases over the last few years.

To find out what the differences are some prototypes were built to see whether one of the languages offers some advantage over the other. In the literature I also looked at some comparisons and reviews of the languages mainly on the internet, but unfortunately a lot of that information is old or very subjective. To compare the two languages the quality characteristics are used to show the differences.

When building the prototypes I noticed some differences already. During the implementation of the Java prototype I sometimes encountered problems where I could not find a solution in the literature. Two examples are given below:

- One example was a problem with rights, in the policy I had to give all permissions for authentication. From a security perspective that is something you do not want, you want to give as few permission as possible. That was not possible, even giving all sub permissions did not work, while using the all permission did work. It would be logical to assume that giving all sub permissions is the same as giving all permissions. In the literature I could only find that it should work with setting some specified sub permissions but it did not work. Even using examples from the internet did not work. It might be that it is because of a newer Java version or something like that. When implementing things like this en encountering strange problems like the one described you would want to find some documentation for it, but I could not find it.
- Another thing that took me some time was to setup the Tomcat server and build a working example. That might be partly my fault, but the fact is that with .NET I could build the web example quite easy. For the .NET part there was also more documentation and examples available, Microsoft itself maintains a quite good selection of reference material.

Another issue is the abstraction layer; Microsoft offers membership and role providers which can be used to build an identity management solution independent of the underlying database. That makes it easier to switch the database. JAAS offers some abstraction with the login modules; it is easy to switch from for instance Kerberos authentication to Windows authentication. Especially Microsoft is busy with these extra abstraction layers, the membership and role providers are added in recent years. It makes it a bit easier for the programmer as the code remains mostly the same while the resources they access can change. It can be seen as offering some extra support, making it easier to setup and maintain applications. Programmers no longer need to write these abstraction levels themselves, it is just build in. One remark here is that Microsoft offers these extra abstractions mainly for Microsoft products like SQL Server and Active Directory while Java offers support for many different products. It is possible to add the support to .NET but that is up to the programmer, in that aspect Java offers more support for other products. Java can be easier to use then .NET when you want to connect to non-Microsoft products as the support for other products is better.

With the quality aspects the differences between JAAS and .NET can be described as follows:

Functionality

- With both Java and .NET the same functionality can be achieved. The examples have shown that it is possible to built authentication and authorization in Java and .NET for both standard applications and web applications. However both offer some levels of abstraction which can make life easier. Java offers login modules and policy files while .NET offers membership and role providers. These abstractions make it easier to implement authentication and authorization, especially since there are some standard modules and providers available.
- Java offers more support for other products with for instance the login module which you can use to easily switch between Kerberos and Windows authentication for example. The .NET framework offers mainly support for Microsoft products like Active Directory and SQL Server with the membership and role providers. The advantage for .NET is that if you use (mainly) Microsoft products that the integration is better since .NET offers lots of support and abstraction layers for their own products.
- Other vendors than SUN offer their implementations based on Java and offer extensions to operate with different products. Microsoft is more or less the only big contributor to .NET.
- For Java there are multiple (paid and free) web and application servers and Integrated Development Environment's (IDE) available. For .NET you have to stick with Microsoft's IIS and Visual Studio.
- Some people say that the GUI from .NET is better looking than the GUI from Java, but that might be a bit subjective.

Reliability

- The Java and .NET platform are quite mature and although there is some debate on their reliability it is mainly a discussion between so called fan boys. In reality there should not be too much difference between the two.

Usability

- Microsoft offers some good documentation including examples on their site. The documentation is well maintained and for every new .NET version there is new documentation. Since most of the documentation can be found on the Microsoft site it is easy to search for if you encounter any problem. Apart from the Microsoft site there is also lots of information available on the internet and in books. That documentation makes it easy to setup applications, maintain them and search for solutions when you encounter problems. Java also has some documentation available, but less then Microsoft. The documentation for Java is also spread over numerous sites and forums which makes it harder to search for information. The amount of information makes it probably easier to learn and understand .NET then to learn and understand Java/JAAS.

Efficiency

- Both Java and .NET require more or less the same resources. However to use all functionality from .NET it is required that you have a Microsoft platform available.

Maintainability

- Hard to say some concrete things here, both Java and .NET are quite mature and good to maintain. The .NET platform might be a bit easier to maintain thanks to the available documentation.

Portability

- When switching from products, for instance when switching from Microsoft products to open source. Java might be the better choice in that case, since it offers better support for multiple products.
- Java supports multiple operating systems, which makes switching easy. Microsoft's .NET can be used to some extend on other operating systems with Mono, but that is limited. Mono is for instance not able to deal with specific Windows only security.

In the end both platform offer the same functionality, so they can both be seen as good choices. The main positive facts for Java are the availability of much free software, like IDE's and servers. Another important thing is that Java runs on multiple platforms and offers better support for other products. The main positive facts for .NET are the available documentation, the support for Microsoft products and the level of abstraction.

It is hard to say which platform you should choose. If you want to use free software go for Java. If you have a Microsoft network it might be the easiest way to use .NET. If you think that portability and interoperability is important then Java might be a better choice. Java offers better support for third party products. In the end the most important thing is that you can achieve what you want with both platforms. It might be best to just look at the skills and experience that your IT staff has. If they already work mainly with Java or .NET, then it is wiser to stick with that choice. If not you might follow the platform that is mainly used within your organization.

The conclusion seems a bit obvious as it was expectable that both Java and .NET would not differ too much. But as you could see in the previous and following paragraphs it is quite difficult to implement the technical part of identity management. The question is whether Java or .NET are suitable for building identity management from scratch. If identity management should be build from scratch then it might take quite some time and it is a difficult task. That it is not easy to implement identity management is shown in the following chapter about guidelines. Identity management is a big subject and lots of guidelines are available. Maybe it is even to difficult to build an entire identity management system for an organization. It might be a better option to use a standard package that already complies with the guidelines.

10 Guidelines

Setting up identity management in a good way that offers the most benefits to the organization is a difficult task. Not alone for the organization but also for the technical part. As identity management is introduced in lots of organizations over the years quite some information is available on the implementation. That information can be used to make security patterns, best practices, guidelines, list pitfalls or list do's and don'ts. These lists and descriptions can help organizations to implement the technical part of identity management while avoiding common pitfalls.

For every project it is necessary to develop your own steps and use the standards and frameworks that are best applicable to you. The most important thing that needs to be done is to create the awareness that process, policy and technology should work together. Both the management and the administrators have to work together to setup an identity management that is as effective as possible. To help with the implementation of identity management this chapter was added.

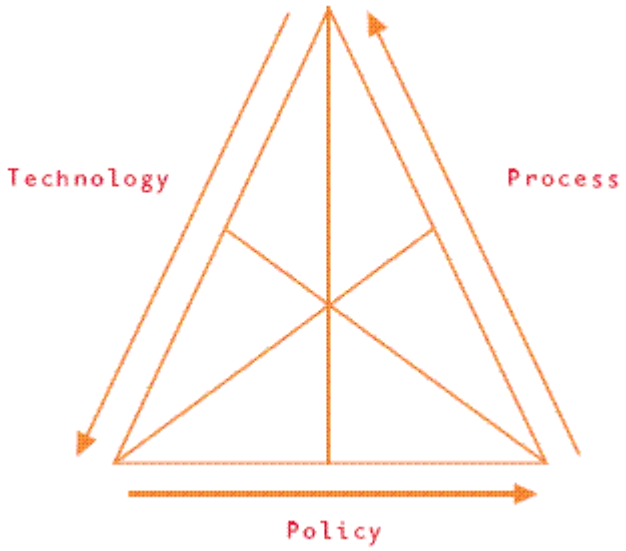


Figure 16 Identity management essential elements (2)

The first two paragraphs from this chapter are about managerial guidelines while the third and fourth are more technical. So the beginning is most interesting for people involved in organizational management while the end is more interesting for programmers/administrators. As this chapter is quite detailed and not really necessary for the understanding of the rest of this thesis it is possible to skip some of the paragraphs.

In this chapter some patterns, best practices and pitfalls are introduced. This will only be a short introduction as there is an excellent book that covers it available. Core Security Patterns (22) is a very helpful book for people implementing identity management, especially if the implementation is in Java as the book is based on Java. In the book the Java language security elements are described followed by lots of patterns, best practices and pitfalls for various topics about identity management. For more specific guidelines for .NET you can have a look at the Microsoft site. To provide a starting point you can have a look at the following topics:

- Patterns & practices Security Guidance for .NET Framework 2.0[21]
- How To: Use Forms Authentication with Active Directory in ASP.NET 2.0[22]
- Active Directory Best practices[23]
- Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication[24]

As Microsoft makes new documentation for new .NET versions it is probably better to check the Microsoft site and search on keywords from the above topics to find the newest documentation.

10.1 Identity management in the organization

Identity management has large consequences, not only for the technical part, but also for the processes within the organizations. Those processes need to be refined to support identity management. Before starting the process to use identity management one needs to find out what the target is and what the scope of the project will be. When determining this one needs to regard all stakeholders: employee/user, managers and administrators.

For the business processes it is a good idea to use AO as a basis for identity management. Another standard that can be used is ISO 27002 (17799). It covers some of the same topics that AO covers but it is a bit more technically orientated. The following issues are covered with ISO 27002[25]: risk assessment, security policy, organization of information security, asset management, human resources security, physical and environmental security, communications and operations management, access control, information system acquisition development and maintenance, information security identity management, business continuity management and compliance.

When you want to setup identity management it is a good idea to not only use something like AO or ISO 27002 but also to use an identity management framework like the one on Figure 17. The framework in the picture displays the key components of identity management in a top-down approach where each layer relies on the layer above. The framework can be used to make sure that the eventual implementation is based on the business processes to prevent that the implementation will be ad-hoc. The framework should help align identity management to the organizations business goals and security strategy.

An identity management framework helps align identity management initiatives with the organization's business goals and security strategy. It also focuses on issues related to (32):

- Delivering business value
- Data confidentiality and integrity
- Nonrepudiation
- Authentication and authorization
- Provisioning and deprovisioning
- Auditing
- Compliance and monitoring

[21] http://msdn.microsoft.com/en-us/library/ms954725.aspx
[22] http://msdn.microsoft.com/en-us/library/ms998360.aspx
[23] http://technet2.microsoft.com/windowsserver/en/library/5712b108-176a-4592-bcde-a61e733579301033.mspx?mfr=true
[24] http://msdn.microsoft.com/en-us/library/aa302415.aspx
[25] For more information see: http://en.wikipedia.org/wiki/ISO_17799

In Figure 17 you can see an identity management framework where the top is the starting point. The organization should develop a security vision first and with the help of the steps in the framework they will end with an identity management roadmap.

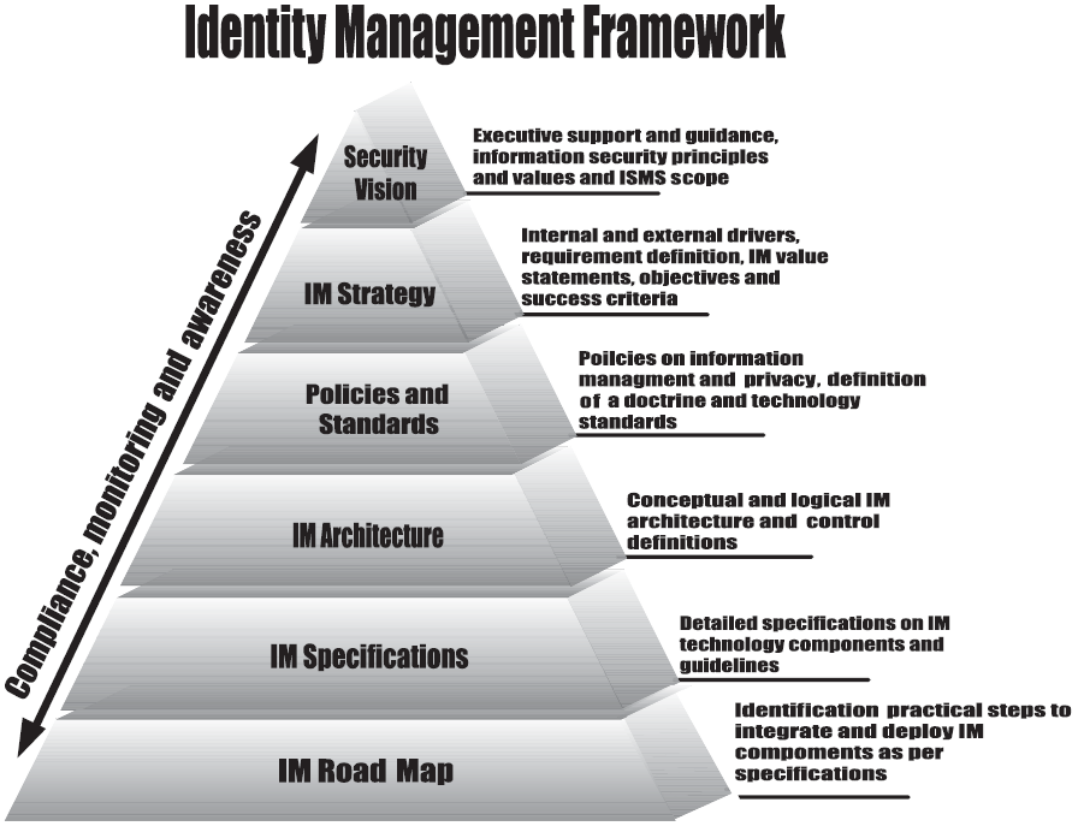


Figure 17 Identity management framework (32)

10.2 Identity management path

The standards together with the frameworks and list of key issues should enable an organization to introduce an effective identity management solution. To keep track of the introduction of identity management it is necessary to define the stages in the process. Therefore I found two slightly different approaches who should give you a basis when selecting your own path. There is no universal path, some companies are only interested in complying with the law, and others want to introduce identity management to maximize business benefits. Every company should set its own path where elements from the paths below can be used as a basis. This paragraph is more meant as a reference just as the previous paragraph, it is possible to skip the remainder of this paragraph and continue after it.

One set of steps or guidelines that are necessary to introduce identity management according to Van Staaij (33):

- Define the business case including budget
- Initial phase
    - Define and prioritize issues and problems
    - Define surroundings
        - Stakeholders
        - Relevant company processes
        - Organization structure
        - Technical infrastructure

        - Look at laws and regulations
- Architectural phase (functional design)
    - Describe architectural principles
    - Description of processes
    - Determine necessary identity data
    - Determine necessary identity management systems
- Product selection
    - Determine selection criteria with stakeholders
- Realization
    - Technical design
    - Development
    - Testing
    - Transfer to maintenance
    - Training
    - Acceptation test

Another set of steps that are identified in order to setup identity management in a successful way are (21):

- Determine project sponsor
    - Interested party and budget keeper
    - Responsible
- Determine primary target of Identity management
    - Centralized maintenance
    - Single sign on (user convenience)
    - Federated identity
    - Compliance to rules and legislation
- Costs
    - Cost containment should not be the only drives as it is difficult to make all costs explicit
- Make a list of all information systems and user databases
- Determine the scenario based on the answers to the previous steps
- Determine the scope (for instance only identity management no code cleanup)
- Arrange project
    - Processes
        - Clearly stated authorization request process
    - Implementation

As every organization is unique and the requirements differ between organizations it is important to set up a specific path for every organization. The lists above can help as guidance to construct that list.

10.3 Laws of identity and other guidelines

In Core Security Patterns (22) many guidelines are discussed, so that is an excellent overview. Except for the guidelines in the book there are some other interesting ones that I would like to include in this paper. These guidelines are probably only interesting when you are actually implementing the identity management, others might want to skip this section.

Some high level design guidelines for identity management are published by Microsoft (34) and should be followed when introducing identity management within a company. These laws are not strict but serve as guidelines. The laws are constructed based on an analysis of identity management. The laws are already quite common as lots of organizations and software houses use them. It is a good idea to follow these rules when implementing identity management. These are the "Laws of Identity" (34):

"1. User Control and Consent: *Digital identity systems must only reveal information identifying a user with the user's consent.*

2. Limited Disclosure for Limited Use: *The solution which discloses the least identifying information and best limits its use is the most stable, long-term solution.*

3. The Law of Fewest Parties: *Digital identity systems must limit disclosure of identifying information to parties having a necessary and justifiable place in a given identity relationship.*

4. Directed Identity: *A universal identity metasystem must support both "omnidirectional" identifiers for use by public entities and "unidirectional" identifiers for private entities, thus facilitating discovery while preventing unnecessary release of correlation handles.*

5. Pluralism of Operators and Technologies: *A universal identity metasystem must channel and enable the interworking of multiple identity technologies run by multiple identity providers.*

6. Human Integration: *A unifying identity metasystem must define the human user as a component integrated through protected and unambiguous human-machine communications.*

7. Consistent Experience Across Contexts: *A unifying identity metasystem must provide a simple consistent experience while enabling separation of contexts through multiple operators and technologies."*

For more information about the laws you can have a look at the Identity blog from Microsoft (34).

There are many more guidelines available for identity management. The following is a list of guidelines that is a bit different than the laws which are discussed above. These guidelines are worth looking at and can be combined with the laws of identity to form a set of guidelines. These key issues should be kept in mind during all stages of identity management implementation (35):

- Think beyond straight identity management: you should not only identify users, look at their roles within the organization, use audit software to cover for gaps in identity management software and use entitlement management where authentication does not stop after you get access to an application, it can manage access to parts within the application.
- Federate and Open Up: use federated identity management to allow identity management across organizational boundaries. It can also support collaboration with business partners. Use open standards like SAML 2.0 to be compliant to other infrastructures. SAML 2.0 also offers a mechanism to control access based on the period of time which is necessary for legislation compliance.
- Put process before technology: identity management is no technical issue; it involves business processes in the first place. Identify the roles within the organization and which authorizations they require. Identity management is no longer just a tool to manage user identifications, nowadays it is also used to enforce compliance and support audit.
- Implement smart: do not build the entire identity management architecture and deploy it. Start small with centralized user provisioning and build upon

that. Strong leadership is also necessary as identity management is important for many aspects of an organization.
- Forget about return on investment (sort of): it is hard if not impossible to make existing costs and cost savings with identity management explicit. Identity management is spread across different organizations and not all benefits are explicitly measureable.

10.4 Architectural patterns

Aside from the Core Security Patterns book there is also a paper available which describes some quite abstract patterns for identity management. In the paper 'Architectural Patterns for Enabling Application Security' (36) patterns are described which can be used to design a system with the ability to implement security later on in the development process. In this paragraph the patterns and their relation are described shortly. For more information you can have a look at the paper.

The following patterns are discussed in the paper:

- Single Access Point: solves the problem of having multiple doors to enter the application by setting up only one way to access the system.
- Check Point: solves the problem of break-in attempts by enforcing a company's security policy to secure against break-in attempts and resulting actions which should be executed when a break-in is detected.
- Roles: solves the problem of managing many user-privilege relations by creating role objects that define the permissions for a group of users.
- Session: solves the problem of sharing non unique values by creating an object with the variables that need to be shared by other objects.
- There are two 'competing' patterns for views, one should be chosen for the application:
    - Full View With Errors: solves the problem of users performing operations which they are not authorized for by showing all options for the user and giving an error when the user tries to perform an action where the user is not entitled for.
    - Limited View: solves the problem of users performing operations which they are not authorized for by showing only those options where the user is authorized for.
- Secure Access Layer: solves the problem of integrating application security with the security of the external systems by building application security around the existing system security. The secure access layer is build on top of the low-level security for communication in and out of the application. So the secure access layer defines how the application security should interact with the underlying security such as operating system security. For resources about low level security issues you can have a look at Applied Cryptography (37).

These patterns can be used when designing a system or you can implement them later one, but that will be much harder to achieve. It is easier to implement them at the beginning, that costs some time but later one it saves a lot of time if you want to introduce identity management. The set of patterns in the paper is still quite abstract; more detailed patterns are available as well and will be discussed later.

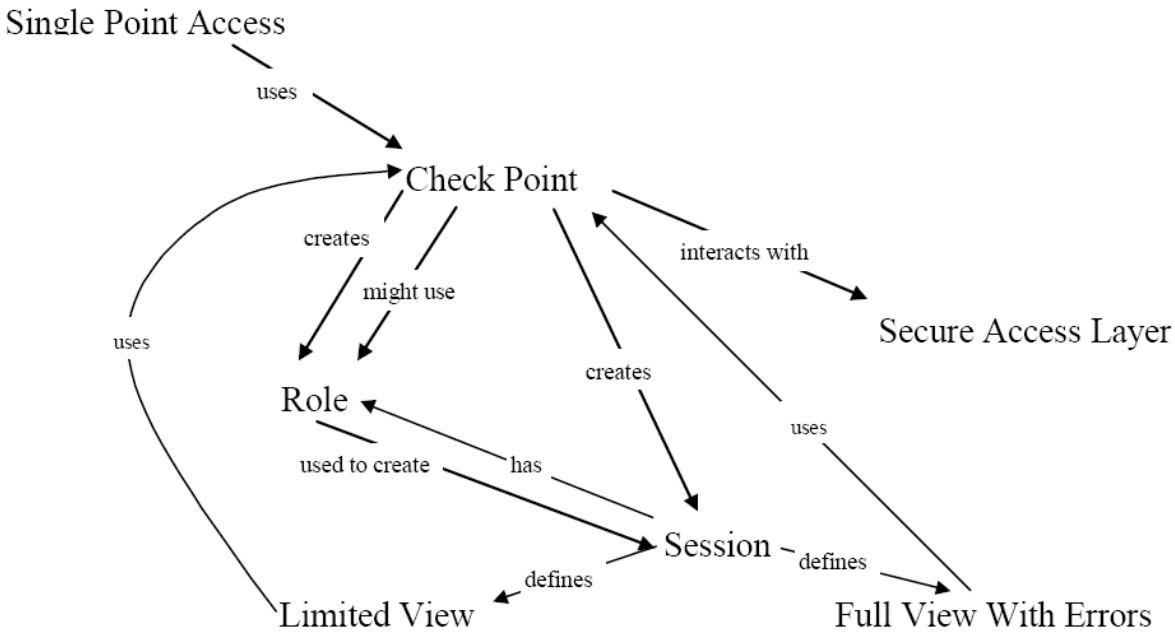


Figure 18 Interaction between patterns(36)

To see how the patterns interact with each other the above picture is included. It is vital that all patterns work together to build a solid base, to see how they can work together you can have a look at the picture. Which actions are performed when a user logs in to the system is described below. It gives you an idea of what actually happens when the patterns are used in practice.

- *Single Access Point* gets the information from the user.
- *Single Access Point* uses *Check Point* which uses the *Secure Access Layer* to validate the user's credentials.
- *Check Point* checks the users *Roles* and then it creates a *Session*.
- The *Session* has a reference to *Role* and can define the *Limited View* and privileges the user has.
- *Check Point* is used by other applications when they need secure operations that cannot be realized at the beginning.
- The system uses a reference to *Session* when it needs global information such as *Roles*, *State* or a *Limited View* of the data.
- *Full View With Errors* or the *Limited Views* interfaces are created using the *Session*.
- The *Check Point* which uses the *Secure Access Layer* can be used to perform security checks while running the application.

10.5 Best practices

The best practices for securing the identity are described in the Core Security Patterns (22) book. These best practices contain advice about what you should do.

- Externalizing Identity Management and Policy Functions:
- Logging and Audit
- High Availability
- SAML Single Sign-on Profile
- Security Threat Mitigation
- Strong Password Policy

Other best practices about securing the Web Tier, the Business Tier, Web Services and secure service Provisioning can be found in the book. And the process of putting everything together is treated as well.

10.6 Pitfalls

The pitfalls for securing the identity are described in the Core Security Patterns (22) book as well. These pitfalls contain information about what you should not do. Some:

- Using Reusable Passwords in Creating Security Tokens
- Using Default Settings
- Using Minimal Security Elements

These pitfalls might be even better then the guidelines since they tell you what you should not do. There are some mistakes that others have made and that should not be made in a new identity management project.

10.7 Conclusion

In this chapter some guidelines, best practices and pitfalls can be found. This is only a small selection. As there is much experience with identity management there is quite some information available. When implementing identity management you should make a selection of guidelines which you want to follow and put them on paper. Following all guidelines is probably not practical, there are simply too many guidelines and complying with all of them might be too much of a hurdle.

These guidelines will not help when deciding to use Java or .NET, as these guidelines are more abstract. Both Java and .NET can be used to implement these guidelines.

11 Conclusion and further research

In this chapter the results of the research are covered. After that there are some recommendations for further research that followed from the research of this thesis.

11.1 Conclusion

The most important conclusion from this thesis is that management, users and administrators should work together. Identity management should be built on a solid foundation based on frameworks and guidelines to build a successful identity management system that can provide as much business benefits as possible. First, one should see identity management as an organizational issue that should be treated as such. The technical part only serves as an implementation to solve the organizational problem. Despite the recent change in view that the organizational aspect is important and the management should not just put the problem in the administrator's hands. It is important that management and administrators work together. If the administrators work together with the management and the users then it will benefit all of them. The management will have more control and can comply with rules and legislation. The administrators can maintain the identity management more easily and will have less work with it. The users have more flexibility and can be more productive (which benefits management as well). Users can change passwords themselves and the processes for requesting authorizations are clearer and rights can be assigned at a faster pace.

The organization should also realize that there are more benefits to be gained from identity management (see chapter 3). One should not just implement identity management to comply with rules and legislation or to reduce costs. Identity management has many more advantages. It can improve the level of security and it can help with auditing as users can be held accountable for their actions. Another benefit of identity management is described above; users, management and administrators need less time to carry out the tasks related to identity management that they normally do. They are also more flexible; users can access the resources they need from outside the organization. For the organization identity management is also very useful when they want to collaborate with suppliers or other organizations. They can give those organizations access to the specific resources that they need.

When the benefits are clear for management then it is time to introduce identity management. Organizations should know what they need to do to implement identity management in a successful way (see chapter 4). That involves not only the technical implementation but also the policies and processes. The policies should state who has access to what, for instance you have employees that have access to enter financial data while others have access to approve that data. The processes define for instance how someone should request for authorization. There the management has to think about issues like who should have access to which resources. The administrators should provide a technical solution to support the organizational ideas that the management has developed.

For that implementation there is lots of help available. There are many guidelines and other documents available. For the actual implementation JAAS and .NET offer some assistance and RBAC can be used to couple organizational roles to 'technical' roles where the authorizations are coupled to. As it is not something simple, one should set up a clear path, use the available standards and look at documentation from the past such as guidelines and best practices (see chapter 10).

To use identity management often some form of database with usernames and passwords is used. Mostly that is done with a so called LDAP server. An application can use the LDAP server to send the username and password entered by the user to authenticate and authorize the user. In this thesis I looked at three different LDAP servers (see chapter 5): OpenLDAP, Fedora Directory Server (FDS) and Active Directory (AD). In practice only FDS and AD are really useful OpenLDAP is a bit hard to install and maintain due to the lack of a graphical user interface. FDS has the main advantages that it is open source and platform independent. AD has the advantages that it has some extra Windows specific features and that it is widely used. Which one to pick depends mainly on the configuration of the network. If there are mostly Windows workstation and servers then AD is probably the best pick. If the network is mostly build with open source products then FDS is probably a better choice. One big issue to keep in mind is that one might have to change from directory server when an organization is merged with another organization. The extra Windows-only functionality AD offers cannot be converted to another directory server so that is something to consider.

When it comes to choosing between different programming languages one will probably have to choose between Java and .NET. These are the most popular languages nowadays. But which of the two is the most useful one when implementing identity management? It appears that both languages are mature enough to implement identity management (see chapter 9). The functionality for identity management that one can achieve with Java and .NET is the same. The difference between the two languages lies on other points. Java has the advantage that it is platform independent, open source and supports multiple software products. Where .NET has the advantage that the documentation is excellent and the support for Microsoft products is very good. That makes it easier to integrate .NET with other Microsoft products like Active Directory. The disadvantage of Java is that the documentation is a bit less than with .NET. The disadvantages of .NET are the platform dependency and the support for third party products is not very well, because they focus on Microsoft product integration.

In the end the differences between the languages are not that big (see chapter 9) that one should choose between one of them just based on the differences. The differences can have an influence when you choose between them, but there are more important factors. One should look at the expertise in the organization, if there is more expertise with one language then the organization should stick with that. Both languages are not very difficult, but it is easier to work with a language you are familiar with. Another factor is the configuration of the organizations network. Are there mainly Microsoft workstations and servers or is the software mainly written in .NET? The latter case it might be a good reason to choose for .NET as it integrates better with the rest. If the network is configured with mainly non Microsoft products like for instance Linux or the programs are mainly written in Java then Java might be a better choice since it offers more support for non Microsoft products then .NET.

The conclusion is that identity management is not something simple but it might be worth introducing it since it has lots of benefits for the organization. The only big disadvantage is the time and money it costs to introduce it. However I believe that it is important that some more research is done. It might be interesting to see an actual comparison of identity management with Java and .NET in an organization. An independent report with issues of actual implementation can help as well. For this thesis only simple applications have been built, they are not comparable to the quite extensive identity management systems within organizations. The documentation that is available now is quite extensive. Complete books are written about identity management. The problem however is that it will be quite hard to actually remember

and use all that information. It would be more useful to have a short list with the most important aspects. However it might be that most companies use standard packages from companies like CA and Sun which include the entire identity management solution. If that is the case then clearer guidelines might not be necessary, so it might be interesting to find out how many companies use standard packages and how many implement identity management themselves.

11.2 Directions for further research

In this paragraph some directions for further research are given. These directions followed from the research that I did. The comparisons were often difficult because there was not much information available or there was no information from an independent organization which makes the comparison quite difficult. The different topics where there was a lack of information available to make a good comparison are treated below.

11.2.1 Directory servers

The information that is available for directory servers is mostly from the companies that implemented them. If it is not from those companies then it is mostly from ardent believers that start flame wars that do not add much useful information to the discussion. Even the comparisons on the internet are mostly not from an independent company or they are outdated. It might be interesting to see some independent research for directory servers that cover the following topics:

- Differences in functionality.
- Ability to port one directory server to another directory server from another company. For instance to check whether it is possible to move a directory server from Active Directory to Fedora Directory Server.

11.2.2 Actual implementation

Although it is quite possible to build identity management from scratch in Java or .NET it is quite a task to let it work with things like federated identities, SOA etcetera. To solve the task of building it yourself there are some packages available from companies like CA, SUN etcetera. It would be interesting to see a comparison between building identity management from scratch and using a standard package. One could make different implementations, one in Java one in .NET and one or more with standard packages. The implementations used in this paper are quite basic and no complete identity management solutions. It would be interesting to compare an actual identity management solution against a standard package. The comparison should then focus on:

- Number of problems encountered in the different implementations. The actual problems should be written down as well, so that they can be used to construct best practices and pitfalls. It is also a good idea to use the encountered problems to build a shortlist of best practices instead of the long lists of practices that are common today.
- Compare Java, .NET and standard packages to see which functionality is already available and what needs to be added.
- The time it costs to implement the entire identity management. So how many hours are made before everything works according to the specification.
- Compare the total costs, so the package costs, the hours that were necessary to implement identity management etcetera. This is important information when an organization has to decide whether to implement identity management itself or to buy a package.
- Collecting data on what is used mainly in companies, a standard package or do they implement their own identity management solution.

The issues above should provide quite some useful information for advanced identity management. The examples in this thesis are simple authentication and authorization, in practice an identity management system is more complex. The information from the research can then be used in the future for organizations that want to introduce identity management. As stated earlier it might not be a good idea to choose between Java and .NET because standard package might prove more value for money.

12 Bibliography

1. **Lewis, Jamie.** *Enterprise Identity Management: It's About the Business.* s.l. : The Burton Group Directory and Security Strategies Report, 2003.
2. **Chong, Frederick.** The Architecture Journal. *Identity and Access Management.* [Online] Microsoft, July 2004. [Cited: March 31, 2008.] http://msdn2.microsoft.com/en-us/library/aa480030.aspx.
3. **Dr. ir. J.M. Bots, Ir. E. van Heck, Drs. V. van Swede, Prof. dr. ir. J.L. Simons.** *Bestuurlijke Informatiekunde in kort bestek.* Rijswijk : Cap Gemini Publishing B.V., 1992.
4. **Staaij, Rob van der.** Medewerkers hebben vaak overbodige rechten. *Automatisering Gids.* April 28, 2006, Vol. 17.
5. **Swensen, Todd.** Building the Business Case for Secure Identity Management. *Connection Magazine.* [Online] Novell, Februari 2003. [Cited: March 28, 2008.] http://www.novell.com/connectionmagazine/2003/02/bottom_line.html.
6. **Quest Software.** *2008 identity management survey.* s.l. : Quest Software, 2008.
7. Identity Management Software Market to Reach $4.9 Billion by 2012, According to New Report by Global Industry Analysts, Inc. [Online] 2008. [Cited: 5 15, 2008.] http://pdfserver.prweb.com/pdfdownload/900984/pr.pdf.
8. **M-Tech.** http://mtechit.com/. *Regulatory Compliance Using Identity Management.* [Online] M-Tech. [Cited: March 28, 2008.] http://mtechit.com/compliance/regulatory-compliance-using-identity-management.html.
9. **KPMG.** *Identity management survey 2003.* s.l. : KPMG, 2003.
10. Talking Identity. [Online] Oracle, 5 2, 2007. [Cited: 5 14, 2008.] http://blogs.oracle.com/talkingidentity/newsItems/departments/oracleidentitymanagement/2007/05/02.
11. *The Architecture Journal: Identity and Access Management.* **Chong, Frederick.** 7, s.l. : Microsoft Corporation, 2004. http://msdn2.microsoft.com/en-us/library/aa480030.aspx.
12. **Jan van Praat, Hans Suerink.** *Bestuurlijke informatiekunde: Inleiding EDP-Auditing.* Deventer : Kluwer Bedrijswetenschappen, 1992.
13. **Rijsenbrij, Prof. dr. D.B.B.** Automatisering van de informatievoorziening. *Informatiebeleid, -planning en -architectuur.* [Online] [Cited: 2 25, 2008.] http://home.hetnet.nl/~daanrijsenbrij/ebi/nl/h9.htm.
14. **Peter Valkenburg, Peter Jurg.** *Identity management: Omgaan met elektronische identiteiten.* Den Haag : Sdu Uitgevers, 2007.
15. **Bemelmans, Prof Dr. T.M.A.** *Bestuurlijke informatiesystemen en automarisering.* s.l. : Kluwer Bedrijfsinformatie, 1998.
16. **Jans.** *Grondslagen administratieve organisatie.* Groningen : Wolters-Noordhoff, 2001.
17. **Ahuja, Jay.** Identity Management: A Business Strategy for Collaborative Commerce. *Information Systems Control Journal.* 2003, Vol. 5.
18. **IBM.** Identity management. *First line of defence for your information assets.* [Online] IBM. [Cited: March 28, 2008.] http://www-07.ibm.com/software/sg/offers/identity_management/index.html.
19. **James A. O'Brien, George M. Marakas.** *Management information systems.* New York : McGraw-Hill, 2006.
20. **Sun Microsystems.** The Business Case for Identity Management. *A Business White Paper.* September 2004.
21. **Bosch, Robbert.** *Handreiking Identiteitenbeheer.* s.l. : GVIB, 2005.
22. **Christopher Steel, Ramesh Nagappan, Ray Lai.** *Core Security Patterns: Best Practices and Strategies for J2EE, Web Services, and Identity Management.* Upper Saddle River : Pearson Education, 2006. 0131463071.

23. *Proposed NIST Standard for Role-Based Access Control.* **D.F.Ferraiolo.** 3, s.l. : ACM Transactions on Information and System Security, 2001, Vol. 4.
24. **Engelbert, Paul.** Five Keys to a Successful Identity and Access Management Implementation. [Online] 2008. [Cited: 5 15, 2008.] http://ca.com/us/eitm/collateral.aspx?cid=172357.
25. **CA.** Services Brief: CA Identity and Access Management Services. *Leverage CA Services Best Practices to Implement an Effective Identity and Access Management Solution.* [Online] 2008. [Cited: 5 15, 2008.] http://ca.com/Files/ServicesBriefs/iam_services_brief.pdf.
26. **Kaushik, Nishant.** Externalizing Identity. [Online] [Cited: 5 20, 2008.] http://static7.userland.com/oracle/gems/nishantKaushik/IDaaSDIDW.pdf.
27. **Saltzer, Jerome H & Schroeder, Michael D.** The Protection of Information in Computer Systems. *Proceedings of the IEEE 63.* 9, 1975.
28. **Bishop, Matt.** *Computer Security: Art and Science.* Boston : Addison-Wesley, 2003.
29. *Java Authentication and Authorization Service (JAAS): Reference Guide.* [Online] Sun. [Cited: 2 29, 2008.] http://java.sun.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html.
30. **Norbert Bieberstein, Sanjay Bose, Marc Fiammante, Keith Jones, Rawn Shah.** *Service-Oriented Architecture Compass.* s.l. : FT Press, 2005. 0131870025.
31. *Identity as a Service - Towards a Service-Oriented Identity Management Architecture.* **Christian Emig, Frank Brandt, Sebastian Kreuzer and Sebastian Abeck.** Karlsruhe : Springer-Verlag, 2007, Vol. LNCS 4606.
32. **Vanamali, Srinivasan.** Identity management framework: Delivering value for business. *Information systems control journal.* 2004, Vol. 4.
33. **Staaij, Rob van der.** Rob van der Staaij. *Identity management.* [Online] [Cited: March 31, 2008.] http://www.robvanderstaaij.nl/blog/static.php?page=static070211-205058.
34. **Cameron, Kim.** Introduction to the laws of identity. *Identity Weblog.* [Online] January 8, 2006. [Cited: March 31, 2008.] http://www.identityblog.com/?p=354.
35. **Neff, Todd.** *Take Five: Keys to identity management.* s.l. : Compliance Week, 2008.
36. **Joseph Yoder, Jeffrey Barcalow.** *Architectural Patterns for Enabling Application Security.* 1998.
37. **Schneier, Bruce.** *Applied Cryptography 2nd edition.* s.l. : John Wiley and Sons, 1995.