Scientia
Interdisciplinary Studies in Computer Science
17(2): 75-87, July/December 2006

INTERGIs - A proposal of an architecture for the integration of identity management systems

Fabio Raphael Sobiecki de Sales[1], Adilson Eduardo Guelfi[2], Volnys Borges Bernal[2]

[1] Novell do Brasil Software Ltda.
ZIP CODE 04551-060 – São Paulo – SP

[2] Departamento de Engenharia Eletrônica, Escola Politécnica
Universidade de São Paulo – ZIP CODE 05508-900 – São Paulo – SP

fsales@gmail.com, guelfi@lsi.usp.br, volnys@lsi.usp.br

Abstract

The goal of this work is to propose an integration architecture that deals with different Identity Management Systems. Using a flexible creation rule, each database administrator will be able to define the level of integration in which the programs of a company will work using another company's user database. Beyond this main function, this architecture will be able to assist the authentication and access control configuration of local programs. The architecture has an administration web interface, where it is possible to manage programs that use the integration data. The integration architecture, called interGIs, has flexibility, compatibility, performance and safety in integration among different identity management systems.

KEYWORDS: identity management, authentication, authorization, integration.

1 Introduction

Now, companies have been spending time and resources in identity management technologies due to laws or Information Technology security requirements. In order to attend this demand, technology companies developed Identity Management systems (IdM).

Digital Identity has the same function of the identity documents, to identify people by documents issued by governmental offices. These documents have information about the person, like: Name, Parent's Name, Photo and Finger Print. The Digital Identity is an electronic method to identify a user through a reliable information system and then to determine his access level.

IdM's are responsible for managing and storing identities. Under an IdM, it is possible to authenticate, to authorize and to audit user accesses.

Blum (2005) defines identity management in the following way:

"Identity Management is a set of process and supporting infrastructure for the creation, maintenance and use of digital identities."

This administrative approach has a good acceptance in daily use because, with IdM, the network administrator can create only one user, and each company program that needs authentication can go directly to this system and validate the user's credentials.

With IdM evolution, new functionalities appeared, however, the most important one is the User Authorization to programs. This functionality is important because it gives access to the process, and it is more secure when it is executed in a centralized tool.

The use of LDAP – Lightweight Directory Access Protocol (Wahl *et al.*, 1997) is common in IdM implementations. LDAP is a hierarchical database and a set of protocols and data manipulation commands by network. This technology is in accordance with RFC 2251 (Wahl *et al.*, 1997). But this directory can be used by only one user administrative domain.

Nowadays, the integration between two domains is the main challenge for LDAP technology. For example: when two companies merged, a specific user may want to use programs from his native company and from the new company. This task is complex and has high security risk, because any mistake can result in an unauthorized access to confidential information.

The goal of this paper is to develop integration architecture among IdMs so that programs can authenticate and authorize users from any database in a transparent way.

This paper is organized as following: section II presents a discussion about related works that deal with remote authentication and authorization. After that, in section III, we present the prior concepts on: architecture, requirements, components and function. In section IV, a study of scenarios is presented to show examples of how this integration is used. Section V describes some features of integration server related to management, event logs and security issues. Conclusions about this work are presented in section VI.

2 Related Works

Silvestre and Rodriguez (2005) describe a domain integration module, based on the authentication tickets and role base access control defined on source domain, like presented on Shibboleth Model (Carmody, 2001).

In accordance with this work, the user can be authenticated in a source LDAP server, and this LDAP server issues a ticket that contains the user role. This ticket is digitally signed and the signature will be verified by the destination LDAP. Based on the role issued by the source LDAP server, the access levels are defined to the local programs.

We can point out, in this work, the use of role base access to define access levels to a remote user in a local program. It is a quick way to identify an access level because system administrators won't need to configure long and complex access control lists.

But this work doesn't inform in which way the program can communicate with the source LDAP server in a secure way. By default, LDAP transfers data in clear text. Other relevant factor is that both companies need to have equal role definitions or, at least, equivalent roles so that the local system can understand the access level defined to a remote user. It isn't clear the method where roles are matched.

In our work, the identity is always authenticated in the source base. Only XML (eXtensible Markup Language) documents are transferred by secure channels. The integration rule system is dynamic, and allows administrators to establish search filters that identify authorized users.

Jones (2006) describes a web authentication solution based on information cards and an integration system for different authentication and control technologies.

These information cards are like digital certificates, but store data and a specific cryptographic key pair for session control and authentication between the user and the program. Data stored in the information card is exclusive about the card owner, like address, birth date, membership number, position and department.

This is a metasystem that authenticate users using several technologies, such as: Kerberos, Liberty, X.509 or SAML. Beyond this interface with client, the metasystem is based on the Web Services technology. Programs consume client authentications through this Web Service.

Authentication is also treated in a different way. In order to prevent phishing scam attacks, both site and user authenticate each other. These authentications use the information cards that contain specific data and a unique cryptographic key pair of the involved parts. With this cryptographic key pair, the authentication will execute only in the allowed site by the information card and other site cannot reuse the same card.

The importance of this work is site-to-user and user-to-site authentications based on the information card. Resources threatened by attacks like user/password are not used anymore. Moreover, the user feels more secure because can trust in the site that authenticates itself.

Information cards can also provide data related to subject, being it the user or the host. In the case of hosts, host attributes can be loaded into the card through extensions. For example, a specific host extension contained in an information card could be a bitmap image that would be showed to the user whenever it requests access to the host.

The author's approach in this work is specific for web sites. Moreover, this work neither presents the concepts of federation or authentication domain integration nor a point where the administrator can configure access rules for a specific user.

In our work, we want to achieve an architecture that is compatible with the present most used programming languages. Moreover, we want to provide a service that can be used by web and client/server programs, hardware and other devices that are requested to be network compatible. The use of digital certificates for authentication can be applied, but this is an administrator's decision.

Santin *et al.* (2002) have developed a work that applies the concept of distributed public keys, SDSI/SPKI (Lampson and Rivest, 1996; Ellison *et al.*, 1999), to establish reliable networks among the provided services.

Through this concept, the user bank belongs to bank customer federation (BCF). A credit card company has a credit card customer federation (CCCF). Between these two federations, a reliable relationship is established.

After that, the entity responsible for BCF will choose an administrator, who will issue nominal certificates to identify its customers, like a Certificate Authority in X.509 standard. The administrator in a SDSI/SPKI specification is responsible for determining and issuing authorization certificates as described in Table 1.

In this way, CCCF another principal will establish and issue name certificates to deliverable services by credit card companies.

In Table 2, we can see the name of the certificate structure with its attributes and its descriptions.

Applying the concept of Network Federation, an electronic commerce company could be considered reliable by its electronic commerce federation (ECF) before the credit card federation. Then, a bank customer, from BCF, can access ECF that supports the electronic commerce functions, by a mutual reliable relationship with CCCF.

For a bank customer to authenticate and to use credit card services, the BCF administrator provides an authorization certificate that lists the rights that has been given by the BCF administrator.

The authentication is done through digital signatures that give validity through the public key issuer transmitted by the reliable network. Authorizations are given validity in two phases. At the first phase, customer signs digitally an operation request, and sends it by the reliable way that allows customer to carry out that operation.

The server gives validity to the authorization certificate through a reliable network and issues a single certificate to the customer. It contains the authorization which was delegated to it by the reliable network. Every time the customer requires the same operation again, it should present this certificate.

At the second phase, the authorization policies are local or a result from administrator's concession through a reliable network.

The server's guardian, a service located in the principal, compares the trusted path provided by the customer with the authorization policy and then it decides for giving access or not to the user.

The use of SDSI/SPKI is interesting in the identification process, because it is a secure method for authentication. Reliable networks allow a dynamic form of authorization well applied in environments like Internet, where there are a lot of people and it is not possible to analyze case-by-case. Internet users can also be separated into very big groups, what makes the use of reliable networks easier.

Our work has a more granular approach, in which rules can be defined by a user, by a user profile or by a great user group. Our work also has focus on the program in limited environments, in which the administrator needs to justify each integration.

About authentication, our work aims to use various authentication technologies, like user/password pair and the use of multifactor authentication as certificates or biometric.

3 Integration Architecture between Identity Management (INTERGIS)

InterGIs architecture has as its main goal, the secure integration between distinguished IdM.

With InterGIs, users originated from a company can authenticate and use programs of another company, using same identity credentials from their original IdM, in a transparent way to users and to programs.

A – REQUIREMENTS

The following requirements were not considered to build this architecture:

- Transparent integration;
- Programming language compatibility;
- Secure communication;
- Integrity data;
- Flexibility in integration rules.

Transparent integration demands the creation of an architecture that can be used in a live environment. We

Table 1. Authorization certificate structure.

Attributes	**Description**
Issuer	Public Key Issuer
Subject	Public Key or administrator name that receive authorization
Delegation	Logic value to authorize the propagation provided by the administrator
Authorization	Permission provided by the issuer
Validity	Validity in a date and time format

Table 2. Name certificate structure.

Attributes	**Description**
Issuer	Issuer Public Key (principal)
Name	Local user name
Subject	A public key or name to be referenced
Validity	Validity in a date and time format

cannot make big changes in the environment by re-implementation, in order to prevent the legacy systems from running correctly. Integration will have to be the least intrusive in the environment at the implementation time.

When program authenticates a user, company "A" or "B", the authentication engine with InterGIs integration, will be the same for any user, without other service to authenticate foreign users.

InterGIs will need to communicate with many different programs. Therefore, it must be compatible with the main development technologies, like: Java and .Net.

It will need to interact with some programs and databases to exchange information. All this communication must be safe to grant that a person or equipment does not obtain privileged information about authentications and authorizations.

Data integrity is a requisite, which demands guarantee and the received information from any users or InterGIs sources have to be really complete and trustworthy. If there isn't an integrity control, an attacker can manipulate information to achieve restricted privileges.

When two companies merge, there are planned conditions or business rules for the integration. For example: Company "Bravo's" users will be able to use only Human Resources program of "Alfa" company's, and "Alfa" company's users will be able to have access, with administrator privileges, to any program of "Bravo" company's.

In one another integration rule example: "Alfa" company users and "Bravo" company users, who are member of "Auditors" group, will have view access in accounting program of "Alfa" company, until a date previously defined, like on final date of auditors contract.

During the InterGIs architecture introduction, the IdM administrator will be able to specify flexible rules as the ones above, so that at the moment of authentication or integration, the rule takes over.

At this moment, each IdM administrator will determine the users' access level an integrated company will have on its own domain. Integration will do not have to guarantee complete access between the parts, unless it is granted by each company administrator. For example: after an integration, the administrator of "Alfa" company determined that all users of "Bravo" company, who belong to Purchase department, will have access to a specific program. But this configuration made by "Alfa" administrator will run only when "Bravo" administrator allows this query in "Bravo" user database, and then users will be able to use the program at "Alfa" company.

With these requirements, we purpose the following architecture.

B – ARCHITECTURE

For this architecture, we suggest a division in four modules, which will do the Integration Server: Program, Engine, Data Source and Connections.

The PROGRAM module is responsible for the communication between server and programs that will use the services. These programs can be: ERP Systems, CRM Systems, electronic turnstiles, electronic door locks, proxy servers, printers, and VPN services.

The ENGINE module, manages process requests from programs to IdMs. It stores the integration rules for each integrated program to each IdM available. For each new authentication request and access validation the Engine module will perform a test on the configured rules and generate a positive or negative reply about this request.

DATA SOURCE is a module responsible for the communication between the integration servers and each local IdM. When the Engine module needs to query an IdM, it will request it to the Data Source module and this module will query IdM directly, replying to the Engine.

When the Engine module checks if the authentication request is from a foreign user, it forwards the request to the CONNECTION module responsible for the remote server integration. The module, then, sends the request to other integration server to be processed.

Figure 1 shows the components position in the server and the interaction of these modules with the external environment.

The integration server is represented by all the servers that store the four modules.

To establish a communication channel between two IdMs, this work purposes a dynamic integration model, using Web Services technology and secure communication protocol to SOA.

SOA (Services Oriented Architecture) is an architectural style to system development that the main objective is to exchange services through simple coupling between components. A unit is called service provider and the other part consumer of these services (He, 2003).

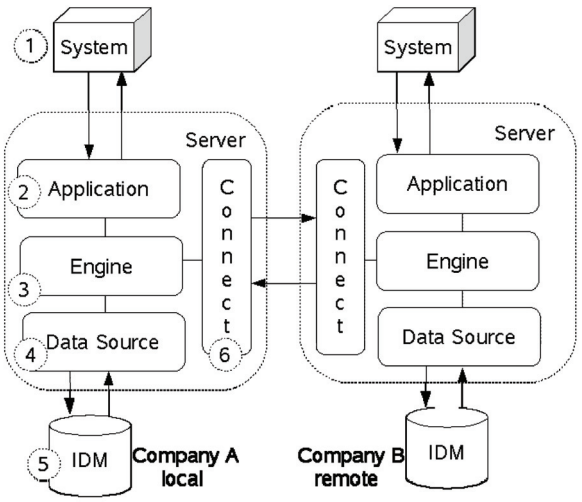

Figure 1. Server components.

This simple coupling, supplied by SOA, is the decisive point to choose this technology. It becomes compatible with existing programs and with new ones, because the majority of the development softwares are compatible with SOA protocol.

The Web Services use SOA to offer services by interoperability standard between different programs running under different platforms and servers (W3C, 2006a).

The option for Web Services as information provider is because the fact that web services are compatible with integration through Internet. It is a technology that has cryptography features for data confidentially. This feature reduces the development of the InterGIs architecture.

Integration server will establish a secure communication between IdM and many programs, through the exchange of XML documents, available in a network service.

XML is a language in flexible text format derived from SGML (ISO 8879). It was originally developed to solve problems of electronic publication in large scale (W3C, 2006b).

B I – APPLICATION MODULE

The main Application module function (item 2 of Figure 1) is to be a channel to programs that will consume data (item 1 of Figure 1). Therefore, it must be compatible with many technologies of hardware and software. Web Services are considered because the way these technologies are able to receive external requests by a server TCP port. The protocol that will be used in this communication is HTTPS to grant the data confidentiality.

A Human Resources (HR) program, which wants to authenticate a user and to authorize it, at a first moment, will collect information about the login name and the user password. After, that program generates an XML document.

After that, HR system requests access to the server through Web Services, where a service is waiting for connections. When connected, it sends the generated Authentication Request XML document.

The Application module, then, will perform a syntax validation on the XML document received. After that, it will interpret the document in order to get the necessary information and also process the request.

In order to process the order, the Application module will redirect the authentication process to the Engine module.

The reply from the Engine module to the Application module will result in a generation of a reply XML document to the HR program. This reply also keeps the syntax constant, as showed in Figure 3.

The administrator server can, if he wants, use a digital signature for each XML document exchanged between the program and the server. However, before it is configured, each Integration Server needs to trust on Certification Authority that issued certificates.

This option of digital signature significantly increases the security, therefore, it prevents spoofing attacks.

B II – DATA SOURCE MODULE

This module is responsible for the connections between the server and the data source (item 4 of Figure 1) that will provide authentication information. The reason for a specific module to perform this task is because each connection is specific to only one database and its implementation must be more optimized for this function.

Only Engine module can request queries to a Data Source. Each Data Source configuration will have a name (line 3, Figure 4) and the same name will be used in the integration rules document.

A main configuration must be informed (line 4, Figure 4). It has to be a LDAP data source with read and write rights. This main data source will be used to store rules of other data sources, programs connections and other related data that each server can run.

We can see, in Figure 4, the data source configuration syntax document.

B III – CONNECTION MODULE

This module is responsible for the connection between the integration servers and for integration between companies, which are the goal of this work (item 6 of Figure 1).

```
01 <xml>
02   <doctype name="authreq">
03     <id>id</id>
04     <time>dd/mm/yyyy hh:mm:ss</time>
05   </doctype>
06   <program>program</program>
07   <user>login</user>
08   <password>password</password>
09   <module>module</module>
10 <xml>
```

Figure 2. Authentication Request syntax document.

```
01 <xml>
02   <doctype name="authrep">
03     <id>id_of_request</id>
04     <time>dd/mm/yyyy hh:mm:ss</time>
05   </doctype>
06   <program>program</program>
07   <messagecode>result_code</messagecode>
08   <message>result_description</message>
09  <module name="module" value="0|1"/>
10 <xml>
```

Figure 3. Authentication Reply syntax.

```
01 <xml>
02   <doctype name="source">
03     <source name="datasource">
04       <main/>
05       <type>ldap</type>
06       <host>server_dns_name</host>
07       <port>tcp_port</port>
08       <user>administrative_account_login</user>
09       <password>password</password>
10     </source>
11   </doctype>
12 <xml>
```

Figure 4. Data Source configuration syntax.

```
01 <xml>
02   <doctype name="exchange">
03     <domain>foreign_domain</domain>
04     <host>server_dns_name</host>
05     <port>port</port>
06     <trustedroot>ssl_certificate</trustedroot>
07   </doctype>
08   <program name="program">
09      <rule name="module" type="ldap">
10          ldap filter
11      </rule>
12   </program>
13 <xml>
```

Figure 5. Connection configuration syntax between servers.

It will receive authentication solicitations from the Engine module, only from foreign users, identified by foreign domain.

The reliable domains are configured in the program rule (line 6, Figure 6). Each reliable domain must have a corresponding configuration in the connection module (line 3, Figure 5).

When the Engine module sends a request, the Connection module transfers the entire document to "Bravo" company's server. The server of company "Bravo's" receives the file by "Bravo's" Connection module.

Therefore, "Bravo's" server must have a program rule, similar to Figure 6, so that "Bravo" server can process requests coming from "Alfa" server.

Before executing the program rule in "Bravo" server, it will give validation to the program rule (lines 14-16, Figure 6) with the configuration of "Alfa" server (lines 8-12, Figure 5).

If there is an inconsistency between this information, the processing is interrupted and an error message is generated.

After "Bravo's" server processes the entire request, the reply of this process returns to the Connection module of "Alfa's" server.

The Connection module returns the entire reply to the Engine module that has already kept a session in progress waiting for the reply. In this way, the local program has only one point of authentication. It is a function for local and remote administrators to allow authentication and authorization.

The Figure 5 shows the connection configuration syntax.

B IV – ENGINE MODULE

It is the main module of the server the responsible for the integration and the processing of authentication solicitations (item 3 of Figure 1).

Based on integration rules, it interprets the request sent by the Program module and its queries to IdM through the Data Source module (item 4 of Figure 1).

When it identifies a foreign user, users that do not belong to the local connected IdM, it redirects the program request from Program module to Connection module (item 6 of Figure 1) after a validation rule that verifies if this operation is permitted. When it receives the reply from the Connection module, it redirects the reply to the Program module without modifications.

The administrator will previously build rules of integration for each program. For example: a human resource program of "Alfa" company that authenticates users in the IdM of "Alfa" company and in the IdM of "Bravo" company.

Figure 6 shows the syntax file for a program to connect on data source trough Intregration Server.

The validation of integration rules runs in the following order:

```
01 <xml>
02   <doctype name="channel">
03     <appl>program</appl>
04   </doctype>
05   <domain>
06    <name>domain</name>
07     <source>data_source</source>
08     <requirements>user</requirements>
09     <requirements>password</requirements>
10     <sourceparam name="data_source">
11       <base>ldapbase</base>
12       <scope>one|sub</scope>
13     </sourceparam>
14     <rule name="module" type="ldap">
15       ldapfilter
16     </rule>
17   </domain>
18 <xml>
```

Figure 6. Syntax of the rules for a application.

a) It verifies if a rule for a program that is requesting authentication exists. This verification uses information contained in XML data of the authentication request. Tag <program> of the request (line 6, Figure 2) must have the same name of the tag <appl> in rule (line 3, Figure 6). If it does not exist, an error message is sent back to the Program module. If the rules exist, they are used to process the request.

b) It verifies if authentication is local or remote, analyzing the tag <user> of the request (line 7, Figure 2). If a domain is different from the specified standard domain, the request is classified as remote; in another case or if it is omitted, a local authentication is assumed. In the remote case, XML request is delivered to the Connection module to be processed in another integration server. Otherwise, if user belongs to the local IdM, the rule is processed with the next step.

c) It verifies if the minimum requirements for authentication have been informed. At the top of XML rule are minimum requirements for the authentication (lines 8 and 9, Figure 6). If there is a compatible rule for a user, like password and email, the Engine module will verify if the authentication request contain these information. Otherwise, an error message will be sent back to the Program module. If all requirements are presented, the module will follow to the next step.

d) It verifies if the user exists. The module will identify in the Engine program rule, which data source is used to verify the authentication of the program (line 7, Figure 6), for that specific domain (line 6, Figure 6). After this, the Engine module will request data from the Data Source, querying if the user informed in the request (line 7, Figure 2) exists and what is the Distinguished Name (DN) for him. DN is the "full name" of object in the LDAP base. This full name verification is necessary to correctly authenticate the user through a LDAP connection. If the query could not obtain a user with the specified login, an error message will be sent back to the Program module.

e) It authenticates the user in the database. The next step, if user exists, is to give validation to the provided password (line 8, Figure 2). The second request sent to the Data Source module, asks for the user authentication using the DN and the password provided by the Application module. Some LDAP server can demand other parameters to authenticate user like search base, search scope. In the IdMs based in a relational database, the information as database name and tables are necessary. This information will consist of the configuration of the rules (lines 10 - 13, Figure 6).
If step "D" results in more than one DN, all the DN's will be given validation and only one positive result will be received. If more than a DN gets positive authentication or in the case of none of the DN's gets success, a message of Program error will be sent to the module.

f) The authorization will be verified. The authorization verification is optional. The developer of the program can choose to verify the user access or not verify it through the server. In small programs that all the users have the same rights, the authorization is something unnecessary. However, in a big program that has many access levels, the validation of authorization through the server is an item recommended for security reasons.
When the developer sends a XML message, informing tags <module> (line 9, Figure 2), it will use an existing module name in the program rule (line 14, Figure 6). The Engine module, then, will look inside the Engine program rule for which requirements the user must fulfill in order to be given validation in the authorization (line 15, Figure 6). For example: in order to have access to the "Financial" module, the user needs to belong to the group called "accounting".
After verifying the necessity of user, the Engine will direct a third query to the Data Source module, to verify if the user fulfills the

specified requirements in the program rule. In positive case, the reply to the program will contain the name of the module with value "1" and in the negative case the reply to the program will contain the name of the module with value "0".

g) The reply is returned to the Program module. At this point, if a document had been directed to the module Connection, the reply of this foreign module is delivered to the Program module. However, if the authentication was local, after all the steps above, a reply with the results of all actions is returned to the Program module.

C - FUNCTIONING

For each company, there will be a server that makes the integration between the data contained in IdM, with the local programs and the remote integration servers.

The communication between the integration server and the IdM will be made using secure protocol LDAP over SSL - Security Socket Layer (Freier *et al.*, 1996). This will grant the data confidentially between the integration server and the data repository.

Beyond the confidentially, the use of LDAPS also grants the authentication between the parts through mutual confidence in the issuer by the certificate of the LDAPS.

An integration server will only make connection with IdMs, which is in its domain. The integration server of company "Bravo" does not connect to the IdM of the "Alfa" company.

When a local user wants to have access to a program of the company, it also will use the infrastructure of the integration server.

After meeting the information necessary to request an authentication, the program writes a XML document, signs it digitally and sends it for the integration server. In the server, the digital signature is verified and the document is processed. The digital signatures grant content authenticity and the authentication of the document sender.

After syntax checking in the Program module, the Engine module will try to locate, amongst the integration rules, one that is addressed to the user domain and to the program specified in the XML on tag <program>.

When the rule is found, the Engine module, will execute the authentication as described. The first query will check the existence of the user. A query is requested by the Engine module to the Data Source module, so that specified login will be located. The query reply, if the login exists, is the DN - Distinguished Name. But, if user does not exist, the authentication process is aborted and the Application module returns an error message to the program that requested the authentication.

With the DN, the Engine module requests the second query to authenticate. The requested query is a connection to the base using the credentials provided in by the Authentication XML document. This connection is made by the Data Source module in the base. If the connection is established, the user is authenticated and the process continues in the next step. However, if it fails, the error aborts the authentication process and the Program module returns an error to the program that requested the authentication.

The third and last step is the verification of authorization. One more time the Engine module requests a query to the Data Source module, that will use the object as the base of the search and a filter LDAP as validation. In case the query returns the user DN, the user fulfills the conditions needed in the program filter and after it the module will return a message of success in the authorization. As a result, the user will gain access to the program. Otherwise, if the last query does not return user DN, Application module will inform that the user is authenticated, but not authorized.

In the authentication of a foreign user that is from another domain, the process has modifications. The process to request login and password of the user do not change in the program; however, the user informs his login followed by "@" and the domain of his original company.

The process at the Application module is identical to the local process. When the request arrives at the Engine module, Engine identifies that the informed domain is different from the configured standard domain that identifies the authentication of a remote user.

After a checking of syntax in the Program module, the Engine module will try to locate, amongst the rules of integration, one that is addressed to the domain of the specified user and the program specified on tag <program>.

When the integration rule is found, the Engine module will redirect the XML document to Connection module that knows foreign Integration Server configuration, and it will send the request to the other Integration Server.

In the Connection module of the foreign user's original company, the remote module receives, validates the signature and directs it to be processed by the remote Engine module. The processing of the request is identical to the local process, exactly because, at this moment, the authentication for that server is local.

The result of the process is directed from the remote Engine module to the remote Connection that then processes a reply XML document, similar to the XML

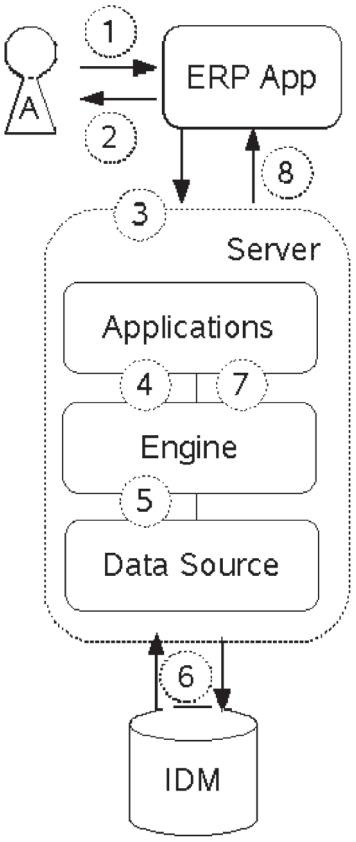

Figure 7. Local authentication.

document issued by the local Application module. If there is a message of error or success, the reply provided by remote server is returned to the local Connection module. The local Connection, then, returns to the local Engine, where it is redirected to the local Application module that, finally, will send an answer to the program that originated all the process.

4 Scenarios

In the study of these scenarios, we will present local and remote authentications.

Figure 7 presents the flow of a company "A" user authentication, authenticating a company "A's" ERP program and using the company "A's" IdM. The flow covers only one server of integration.

1.User request access to the program;

2.Program requests the credentials of access through user and password;

3.Program generates a XML request, digital signs and directs it to the integration server;

```
<xml>
 <doctype name="authreq">
   <id>534</id>
   <time>12/10/2006 08:45:34</time>
 </doctype>
 <program>ERP</program>
 <user>jsilva</user>
 <password>s3cur3#</password>
 <module>Financial</module>
 <module>Logistic</module>
<xml>
```

4. The Program module checks the signature, validates the XML document and redirects to the Engine module;

5.The Engine locates the integration rules and verifies if the program "ERP" can authenticate in the default domain;

```
<xml>
 <doctype name="channel">
   <appl>ERP</appl>
 </doctype>
 <domain>
   <name>a.com.br</name>
   <source>idm-employee</source>
   <requirements>user</requirements>
   <requirements>password</requirements>
   <sourceparam name="idm-employee">
     <base>ou=sao,o=a</base>
     <scope>one</scope>
   </sourceparam>
   <rule name="Sales" type="ldap">
     groupmembership=Sales
   </rule>
   <rule name="Financial" type="ldap">
     aclFinancial=true
   </rule>
   <rule name="Logistic" type="ldap">
     &(groupmembership=Managers)(department=Shipping)
   </rule>
 </domain>
 <domain>
     <name>b.com.br</name>
     <requirements>user</requirements>
     <requirements>password</requirements>
 </domain>
<xml>
```

6. The module data source carries 3 queries in the base: login for DN, authentication of the user and authorization;

```
<xml>
 <doctype name="source">
   <source name="idm-employee">
     <main/>
     <type>ldap</type>
     <host>ldap.a.com.br</host>
     <port>636</port>
     <user>cn=adminint,ou=services,ou=sao,o=a</user>
     <password>t1ck3t32@%</password>
   </source>
 </doctype>
<xml>
```

7. The result of the processing is returned by the program module;

8.The Program module writes an XML document reply, signs and returns to ERP program, granting or not its access.

```
01 <xml>
02  <doctype name="authrep">
03    <id>534</id>
04    <time>12/10/2006 08:45:42</time>
05  </doctype>
06  <program>ERP</program>
07  <messagecode>200</messagecode>
08  <message>User Authenticated</message>
09  <module name="Financial" value="1"/>
10  <module name="Logistic" value="0"/>
11 <xml>
```

Figure 8 demonstrates the flow of an authentication by the "B" company user, who is authenticating in an ERP program of "A" company and using the company "B" IdM. The flow covers two servers of integration now.

1. A user from "B" company requests access to the program;

2. The Program requests the credentials of access through user and password;

3. The Program generates a XML, digital signs and directs to the integration server;

```
01 <xml>
02  <doctype name="authreq">
03    <id>535</id>
04    <time>12/10/2006 08:47:34</time>
05  </doctype>
06  <program>ERP</program>
07  <user>msouza@b.com.br</user>
08  <password>s0ftt3ch</password>
09  <module>Financial</module>
10 <xml>
```

4. The Program module checks the signature, validates the XML and redirect to the Engine module;

5. The Engine locates the integration rules and verifies if the program "ERP" can be authenticated at "B" company's domain and then directs to the Connection module;

```
01 <xml>
02  <doctype name="channel">
03    <appl>ERP</appl>
04  </doctype>
05  <domain>
06    <name>a.com.br</name>
07    <source>idm-employee</source>
08    <requirements>user</requirements>
09    <requirements>password</requirements>
10    <sourceparam name="idm-employee">
11      <base>ou=sao,o=a</base>
12      <scope>one</scope>
13    </sourceparam>
14    <rule name="Comercial" type="ldap">
15      groupmembership=Sales
16    </rule>
17    <rule name="Financeiro" type="ldap">
18      aclFinanceiro=true
19    </rule>
20    <rule name="Logistica" type="ldap">
21      &(groupmembership=Managers)(department=Expedicao)
22    </rule>
23  </domain>
24  <domain>
25      <name>b.com.br</name>
26      <requirements>user</requirements>
27      <requirements>password</requirements>
28  </domain>
29 <xml>
```

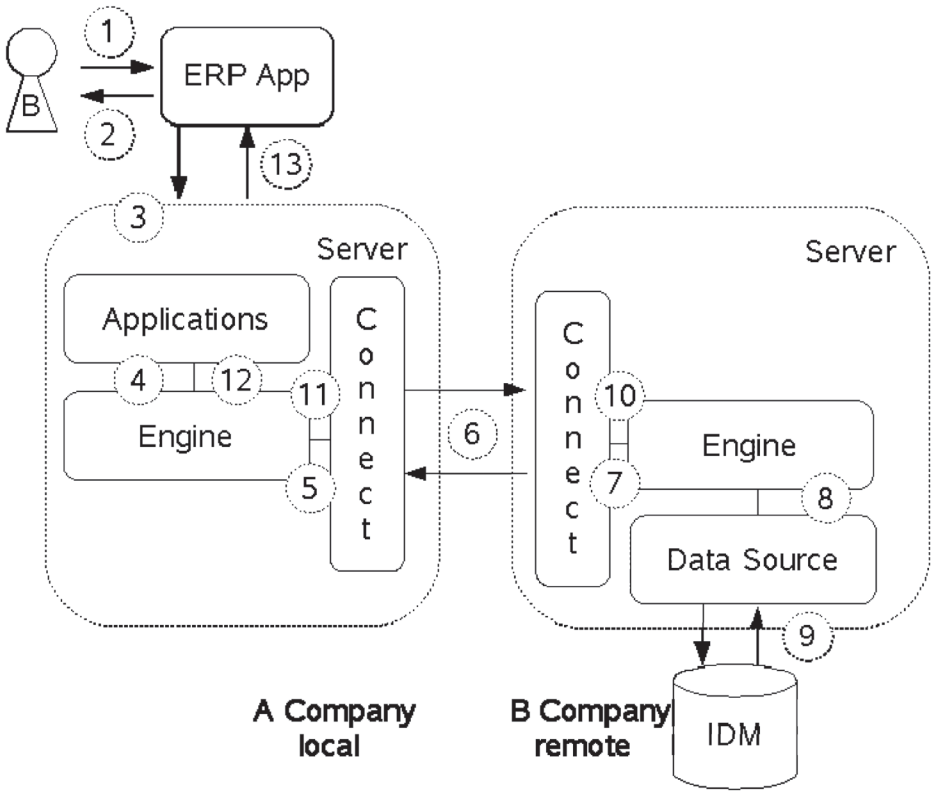

Figure 8. Remote authentication.

6. The Connection module delivers the XML document, containing the original user request and sends it to the remote Connection module through a secure channel;

7. The remote Connection module validates the rule of authorization of the program in company "B", with the rule exported for the Connection module of the "A" Company and directs it to the remote Engine module;

```
01 <xml>
02  <doctype name="exchange">
03    <domain>b.com.br</domain>
04    <host>mickey.b.com.br</host>
05    <port>5110</port>
06    <trustedroot>bcert.der</trustedroot>
07  </doctype>
08  <program name="ERP">
09     <rule name="Financial" type="ldap">
10         groupmembership=Auditors
11     </rule>
12  </program>
13 <xml>
```

8. The Remote Engine module locates the integration rules and verifies if the program "ERP" can authenticate in the default domain;

```
01 <xml>
02  <doctype name="channel">
03    <appl>ERP</appl>
04  </doctype>
05  <domain>
06    <name>b.com.br</name>
07    <source>svwactdir</source>
08    <requirements>user</requirements>
09    <requirements>password</requirements>
10    <sourceparam name="svwactdir">
11      <base>cn=Users,dc=b,dc=com,dc=br</base>
12      <scope>sub</scope>
13    </sourceparam>
14    <rule name="Financial" type="ldap">
15      groupmembership=Auditors
16    </rule>
17 <xml>
```

9. The Data Source module carries the 3 queries in the base: login for dn, authentication of the user and authorization;

```
01 <xml>
02  <doctype name="source">
03    <source name="svwactdir">
04      <main/>
05      <type>ldap</type>
06      <host>svwactdir.b.com.br</host>
07      <port>389</port>
08      <user>cn=administrator,ou=Users,dc=b,dc=com,dc=br</user>
09      <password>do031gbq1zp3%3q0692@d</password>
10    </source>
11  </doctype>
12 <xml>
```

10. The result of the processing is returned to the remote Connection module, that returns it to the local Connection module through the established secure channel in step 6;

```
01 <xml>
02  <doctype name="authrep">
03    <id>535</id>
04    <time>12/10/2006 08:45:43</time>
05  </doctype>
06  <program>ERP</program>
07  <messagecode>200</messagecode>
08  <message>User Authenticated</message>
09  <module name="Financial" value="1"/>
10 <xml>
```

11. The local Connection module returns the received reply from company "B's" integration server, to the local Engine module;

12. The local Engine returns the reply to the Program local module;

13. Finally, the local Program module sends a XML reply to the ERP, with the result of the queries made by the user in company "B", and this grants or not access for the "B" to the program ERP of the company "A".

5 Integration Server Features

A - ADMINISTRATION

Each server will need some administrative actions for its functioning and configuration. For that, an administrative web interface will be developed and can have access through protocol HTTPS.

During the architecture introduction, the administrator will configure the initial parameters of LDAP and will specify a service account that will be used to the queries.

For the first access to the administration interface, the administrator will authenticate with the user and the password of this service account, later through the option "User Management", he will be able to nominate other administrators of the server.

The data sources for LDAP authentication will be configured in the option "Data Sources". It will already access the data in the main connection, which was configured during the installation of the server. Other user data sources can be configured. These sources of data could be LDAP bases as: OpenLDAP, Novell eDirectory, Red Hat Directory, Fedora Directory, Sun Java System Directory or Microsoft Activate Directory. Relational Databases can be configured with the user table as: Oracle, MySQL, DB2, PostgreSQL or others.

However, it is mandatory the main base to be LDAP, therefore, it will keep in this base other structural information of the architecture.

In the "Program" option, programs will be configured to consume the data supplied by the server through the Data Sources.

Any program or hardware that can be connected through HTTPS protocol by net can send XML documents to the server and, if it is in the list of allowed programs, it will receive a document as a reply. In case a request arrives at the server, it will try to match with some rule. In case it does not match it, an XML document error will be returned.

The integration rules will be configured in the "Engine" option. The administrator will configure the rules of integration with the previously configured Data Source.

Finally, in the option of "Records & Logs", the administrator will configure the options to send and store logs.

B - LOGS REGISTER

To track and scan, the server will be able to register logs of executed actions. Events as authentication, access to administrative interface, error messages of connected programs and other system faults will be recorded.

There will be 3 types of messages: "Alert", "Critic" and "Security".

Messages classified as "Alert" are simple messages, only of the informative kind to the administrator, such as: Authentication of one determined program or Access to the administrative interface.

In the classification of "Critic", selected messages must have emergency actions of the administrator. Messages as: Connection fails or Query fails.

The classified messages as being of "Security" will contain related information about attacks against the security. For example: Badly-formed Requests or Successive Authentication Fails.

Logs could be sent to a Syslog server which will be configured to receive the events. Beyond the Syslog, the program will be capable to send the events to a database. The goal is creating a report to analysis of these data and generate systems alert by email, pager or SMS.

C – SECURITY ISSUES

In order to be an integrated architecture of access for the programs that can contain confidential data, the server can be a point of attacks.

To grant the security triad, some actions will have to be analyzed.

Confidentiality - the communications between the server and the data sources or programs need to be protected by secure communication protocols as HTTPS or LDAPS.

Integrity – XML documents have digital signature issued by the program and by the server. Certificates will be issued for each program and a reliable root will be known by hosts.

Availability - As the integration rules remains in the main LDAP, the connection to the LDAP server can be redundant, therefore, the LDAP supports base replication. More than one server can execute the Engine, and a structure of load balance by the hardware can be implemented.

As the server will be waiting for connections in a specific port, badly-formed requests can be directed to this service to try an unavailability of service. The server will have to be capable to detect badly-formed requests and discard them immediately.

6 Conclusions

The Identity Management is part of the Information Security strategy. Because the access control protect restricted information. We must be cautious on how many products are associated to this service.

On the other way, the dynamism of companies requires fast and reliable answers for the most different situations that the company can go under.

One of these changes in companies is the association, mergers or even the services of outsourcing.

In instants, an amount of new identities demand access to the controlled resources of the company. Moreover, the life cycle of this remote identity is extremely variable.

For these reasons, this work was dedicated to develop an architecture that allows, at the same time, flexibility, compatibility, agility and security to the integration between different systems of identity management.

The proposal here achieved these goals, providing architecture based on current and emergent technologies with the focus on the security needed in the technology information environment.

The architecture can be implemented in any type or size of company, once that scalability and conditions of physical support are observed.

Beyond this direct program in company integration, this architecture can have other situations of applicability, for example: an electronic commerce that would like to authenticate, on line, the condition of one customer in social insurance database at the identities database of Federal Government. A second example is a future system of electronic vote, where each vote machine can authenticate and authorize voters in identity bases of Electoral Superior Court.

For the future works, the described architecture can serve as inspiration to an optimized and a more efficient model.

Also, the installation of this architecture in a programming language makes necessary that there be compatibility with the specifications, similar to giving validation to the concepts covered by this work.

Some others technologies of authentication and authorization, like Kerberos, SAML, Liberty Alliance Federation can be integrated with InterGIs to be compatible with other systems.

The integration server could have a cache system to store some integrated identities. At the first moment, the identity is verified by normal process. When the identity is valid, this data will be stored in a secure cache in the integration server. The next query made to same identity, it could be verified by the stored information in the cache.

This is one suggestion for the purpose of this architecture that needs to be planned and implemented in a testing environment.

References

BLUM, D. 2005. Concepts and Definitions: Identity and Privacy Strategies. *In-Depth. Research Overview*. Burton Group, Version 2, p. 12.

CARMODY, S. 2001. Shibboleth overview and requirements. *Shibboleth Working Group*. Available at http://shibboleth.internet2.edu. Accessed on 12/03/2006.

ELLISON, C.; FRANTZ, B.; LAMPSON, B.; RIVEST, R.L.; THOMAS, B. and YLONEN, T. 1999. RFC2693 – SPKI Certificate Theory. Internet Society. Available at http://www.ietf.org/rfc/rfc2693.txt Accessed on 12/03/2006.

FREIER, A.O.; KARLTON, P. and KOCHER, P.C. 1996. The SSL Protocol Version 3.0. *Netscape Communication*. Available at http://wp.netscape.com/eng/ssl3/draft302.txt Accessed on 12/03/2006.

HE, H. 2003. What Is Service-Oriented Architecture, XML.com. Available at http://www.xml.com/pub/a/ws/2003/09/30/soa.html. Accessed on 12/03/2006.

JONES, M.B. 2006. The Identity Metasystem: A User-Centric, Inclusive Web Authentication Solution. *In:* W3C WORKSHOP ON TRANSPARENCY AND USABILITY OF WEB AUTHENTICATION, New York City. Available at http://www.w3.org/2005/Security/usability-ws/papers/28-jones-id-metasystem/. Accessed on 12/03/2006.

LAMPSON, B. and RIVEST, R.L. 1996. *A simple Distributed Security Infrastructure*. MIT – Massachussetts Institute of Technology. Available at http://theory.lcs.mit.edu/~cis/sdsi.html. Accessed on 12/03/2006.

SANTIN, A.O.; FRAGA, J.S.; MELLO, E.R. and SIQUEIRA, F. 2002. Um modelo de autorização e autenticação baseado em redes de confiança para sistemas distribuídos de larga escala. *In:* SIMPÓSIO DE SEGURANÇA EM INFORMÁTICA, 6, São José dos Campos, SP. *Anais...* São José dos Campos, CTA/ITA, p. p.101-110.

SILVESTRE, B.O. and RODRIGUEZ, N. 2005. Autenticação e Controle de Acesso em Serviços de Diretórios Multi-institucionais. *In:* SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, Fortaleza, CE, 23. Available at http://www.sbrc2005.ufc.br/_includes/aceitos.php. Accessed on 05/15/2007.

W3C 2006a. Web Services Activity Statement. *World Wide Web Consortium*. Available at http://www.w3c.org/2002/ws/Activity. Accessed on 12/03/2006.

W3C 2006b. Extensible Markup Language. *World Wide Web Consortium*. Available at http://www.w3c.org/XML. Accessed on 12/03/2006.

WAHL, M.; HOWES, T. and KILLE, S. 1997. *RFC2251 Lightweight Directory Program Protocol* (v3). Internet Society. Available at http://www.ietf.org/rfc/rfc2251.txt