Global Information Assurance Certification Paper

SANS Training & GIAC Certification GSEC PRACTICAL ASSIGNMENT V1.4B OPTION2

INTEGRATION OF SINGLE SIGN ON WITHIN THE FRAMEWORK OF AN J2EE ENVIRONMENT IN BANKING FIELD

PHILIPPE GROS, GCIA

Summary:

The goal of this document is within the framework of the GSEC practical to present a solution of Single Sign On in a J2EE environment in a banking field. It will describe initial context and needs (Environment RACF and strong authentification of working stations), a methodology of project management and our choice of architecture (authentification based on RACF, authorization based on LDAP) and all results obtained. Lastly, it will present on going workings, researches and developments.

1. INTRODUCTION

1.1 PRESENTATION OF WHAT EXISTS

1.1.1 Working stations authentification is done by Certificate.

Working stations have an X-509 certificate. This certificate is used to set up a tunnel SSL between client browsers, HTTP servers or application servers. The principle selected is to enforce a mutual authentification between client browsers and servers. This phase of authentification must be supplemented by controlling these certificates against a revocation list. [PKI].

1.1.2 Weak authentification of "User"

User authentification is done by Login & Password.

After authentification of his working station, a user receives a login page proposed by each application for carrying out his login/password. A validity check of his password must be carried out into a security base such as RACF or others according to an application targeted. In the event of a problem, this user must receive an error message.

1.1.3 Access authorization and users rights:

For each application reached, an authorization process is initiated according with their profiles. There is no centralized rights management.

1.2 RISK ANALYSIS AND DESCRIPTION OF NEEDS

- User authentification is needed before accessing to any applicative resources.
- There should be one mode of authentification and that only one mode by session.
- Only one authentification database must be implemented.
- Available applications must implement authorization access based on users' roles and by using a security context in conformity with J2EE standard.
- Some 4-tier architecture is on target.
- Authentification and authorizations suitable from each user profiles will be done on an access portal Web. This portal will return a specific context to each users giving access to only applications to which it have right.
- In order to limit the changes and to be able to proceed quickly with a first version, it was decided to avoid in a first time to use users strong authentification. It is on the other hand a strategic choice to set up this mode

of authentification in a next future version. Only a strong authentification of each working stations and security administrators will be retained for right now. Authentification of each working stations and security administrators will be retained for a first stage.

- These first elements in conformity with [2] (p 36-37) show the needs to integrate Single Sign On into our whole company.
- Working stations should migrate from SNA protocol and a 3270 emulation to a light client such as an HTTP browser running on TCP/IP.

Risk assessment

- AUTHENTIFICATION.: An absence of centralized management for authentification and authorization access to applications is a strong limit of our current architecture. Moreover, there is a considerable risk not to apply or circumvent the security policy such as for passwords' management:, passwords wrote on screens or workspaces, passwords exchanged between people and so on. The fact of having a strong working stations authentification and not for each user is also a risk.
- ACCESS CONTROL: Users access control to applications and information is carried out on a specific discrete basis to each application. There is once again a risk related to the decentralized management of the reference indexes.
- PHYSICAL SECURITY: Environmental technical protections are currently specific to Mainframes placed in protected room according to sate of art ad standards from our profession. However the passage of SNA to IP opens new vulnerabilities such as deny-of services or man on the middle attacks.
- INFORMATION INTEGRITY: Protection against an unauthorized modification of critical information in our mainframe environment is currently controlled. The introduction of UNIX servers and HTTP protocol into a 4-tiers environment increases the risk of attacks.
- CONFIDENTIALITY : Prevention of an unauthorized disclosure of significant information is all the more significant as the passage to HTTP increases the risk of sniffing information.
- AVAILABILITY: An introduction of 4-tier environment increases risks of a quality of service fails. There is thus a strong need of availability.
- AUDIT: Recording all activities allowing the reconstitution of each transaction or processes is essential with our profession. An introduction of some 4-tier architecture increases the difficulty of having some audit files incoherencies.
- NOT REPUDIATION: An authentification of origin and recipient for transactions is not currently a need because this suggested architecture is currently on an Intranet.

2. PROJECT METHODOLOGY USED

This strategic project affects more than 30000 working stations distributed on a vast geographical area.

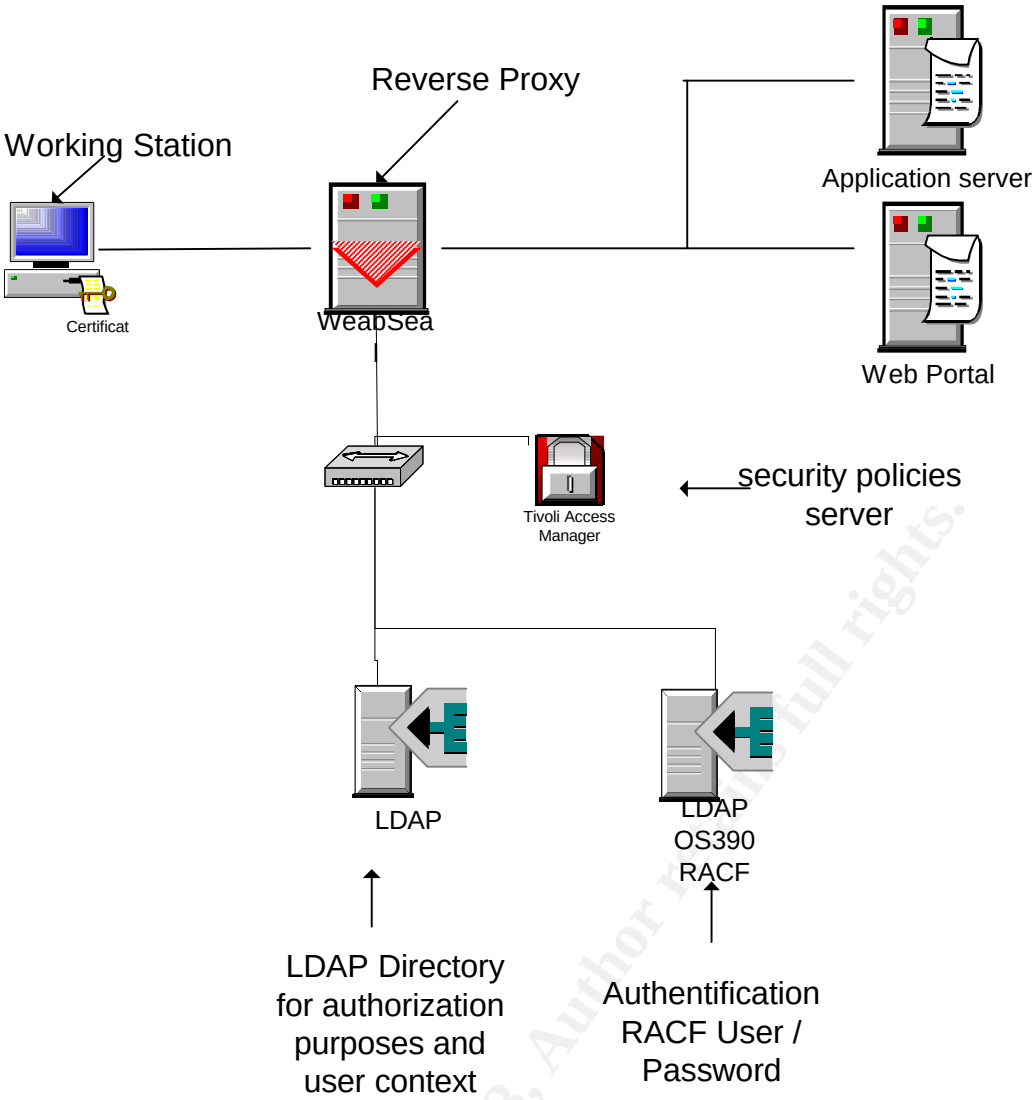
So, a project management methodology for these new kinds of information technologies was set up and was cut out in several phases:

- Feasibility studies and proof-of concept. Theses phases were proceeding in parallel with a RFQ (Request for Quotation). Several manufacturers were solicited and a short list of 2 manufacturers was proposed. They led two prototypes for checking products adequacy.
- Realization of a pilot and finally a first version.
- Deployment.
- Studies of future architecture evolutions and roadmap.

I joined this bank at the time of feasibility studies as senior security architect. My mission was to manage a project team while defining security architecture in depth.

3. SECURITY: AN LOGICAL ARCHITECTURE

This is an N third architecture made-up of 3 levels: a face end (HTTP server), a middle end (application server: WebSphere) and back end (IBM mainframe) Through a Tivoli security server and Websphere application servers are implemented security services.



Here you are our security functional architecture

- **WebSeal Reverse Proxy**: pivot of our architecture Tivoli Access Manager (TAM) is charged to intercept all exchanges issued by client browsers. WebSeal carries out a strong authentification for working stations and a weak authentification for users.
- **A mechanism of cookies** carries out a service of SSO on the whole of the accessible servers.
- **Some programming libraries called CDAS** (Cross Domain Authentication Service) are activated by WebSeal, which were customized to include some specific security controls during the phase of double authentification. (working station and user authentifications)
- **Tivoli Access Manager (TAM) server** itself, which records and manages security policies and control access rules based on identity, groups membership, roles earned and finally an URL targeted for each user.
- **A specific web portal** in order to create a virtual office according to the rights of each user.

- **WebSphere application servers (WAS);**This environment, as well as the Java applications deal with more complex access control functions related to different transactional programs.
- **A RACF security database** for the conservation of users' identifiers and passwords. IBM mainframe uses diverse other security parameters too. Notices: this security database is reached via a LDAP bridge since WebSeal does not know in this implemented version to address directly RACF.
- **A LDAP Directory** for recording some objects exploited by WebSphere and TAM.
- **A certificate management server (PKI)** in charge of distributing and managing certificates for working stations, servers (WebSeal, mainframe, Ldap, WebSphere) and finally security administrators

4 TAM V 3.9 SECURITY SERVER

4.1 FONCTIONNALITIES

Le serveur de sécurité Tivoli Access Manager retenu se compose :

- d'un serveur de règles (« Policy Server ») assurant la gestion des règles de sécurité et des autorisations sur les ressources de l'INTRANET,
- d'un gestionnaire de sécurité chargé de l'application des règles de sécurité : WebSeal.

Il comporte les fonctionnalités suivantes :

- fonctionnement en mode reverse proxy. Il apparaît comme un serveur Web pour les navigateurs et comme un navigateur pour les serveurs Web ou d'applications d'arrière plan.
- protection des serveurs d'applications Web d'arrière plan par le mécanisme de « jonction ».
- prise en charge de plusieurs méthodes d'authentification.
- gestion d'un contrôle d'accès pour l'espace Web du serveur mots et des serveurs d'arrière plan.
- fourniture dynamique d'informations liées aux droits d'accès aux serveurs d'arrière plan qui intègre un mécanisme permettant de transmettre aux serveurs d'arrière plan, dans l'en-tête http, des informations sur les droits d'accès des utilisateurs sous la forme de « variable = valeur ». Les serveurs d'arrière plan extraient ensuite ces données de l'en-tête http sans avoir besoin de mettre en œuvre du code sophistiqué. Ces informations peuvent être utilisées en toute sécurité puisque émanant du serveur de sécurité mots.

Ces informations sont dénommées « contexte sécurité » dans la suite du document.

- capacités de connexion unique (SSO, « Single Sign On ») qui permet à utilisateur d'accéder à une ressource, quel que soit son emplacement, avec uniquement l'authentification liée à la connexion initiale. Ensuite, l'accès à des ressources situées sur des serveurs différents est totalement transparent pour l'utilisateur.

4.2 IDENTIFICATION / AUTHENTIFICATION

4.2.1Principles

[4] Describes various authentification possibilities in a Web world, and how an architecture based on the *SSO* makes it possible to facilitate the management of authentification policies. However, a double authentification implementation was a new challenge which we solved.

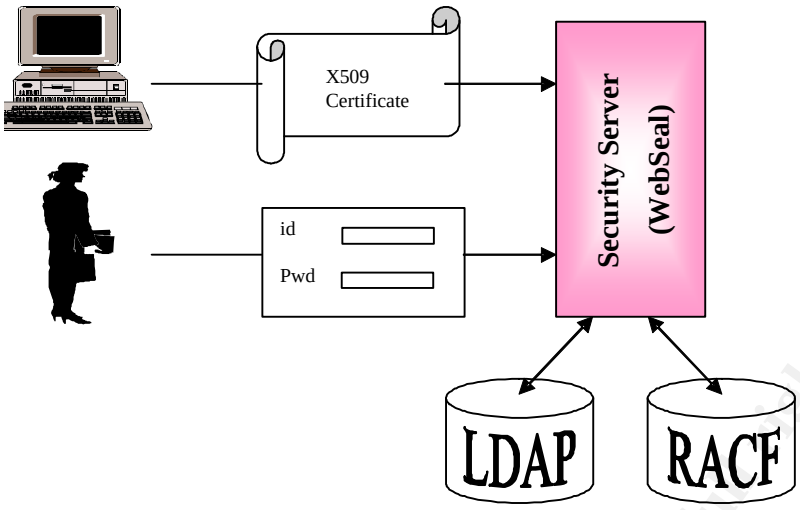
Within the framework of described needs, it was necessary to develop a specific CDAS module to implement a double authentification including:

- A mechanism of strong authentification based on X 509 certificates and a control revocation list.
- A weak authentification mechanism of all users based on an accurate control of user's login/password against our RACF security database.
- A consistency checks in order to verify the good adequacy between working stations, the code entity of the establishment where this user was connected. This operational rule stands to prevent any connection from a user to an application starting from a place not in conformity with his assignment.

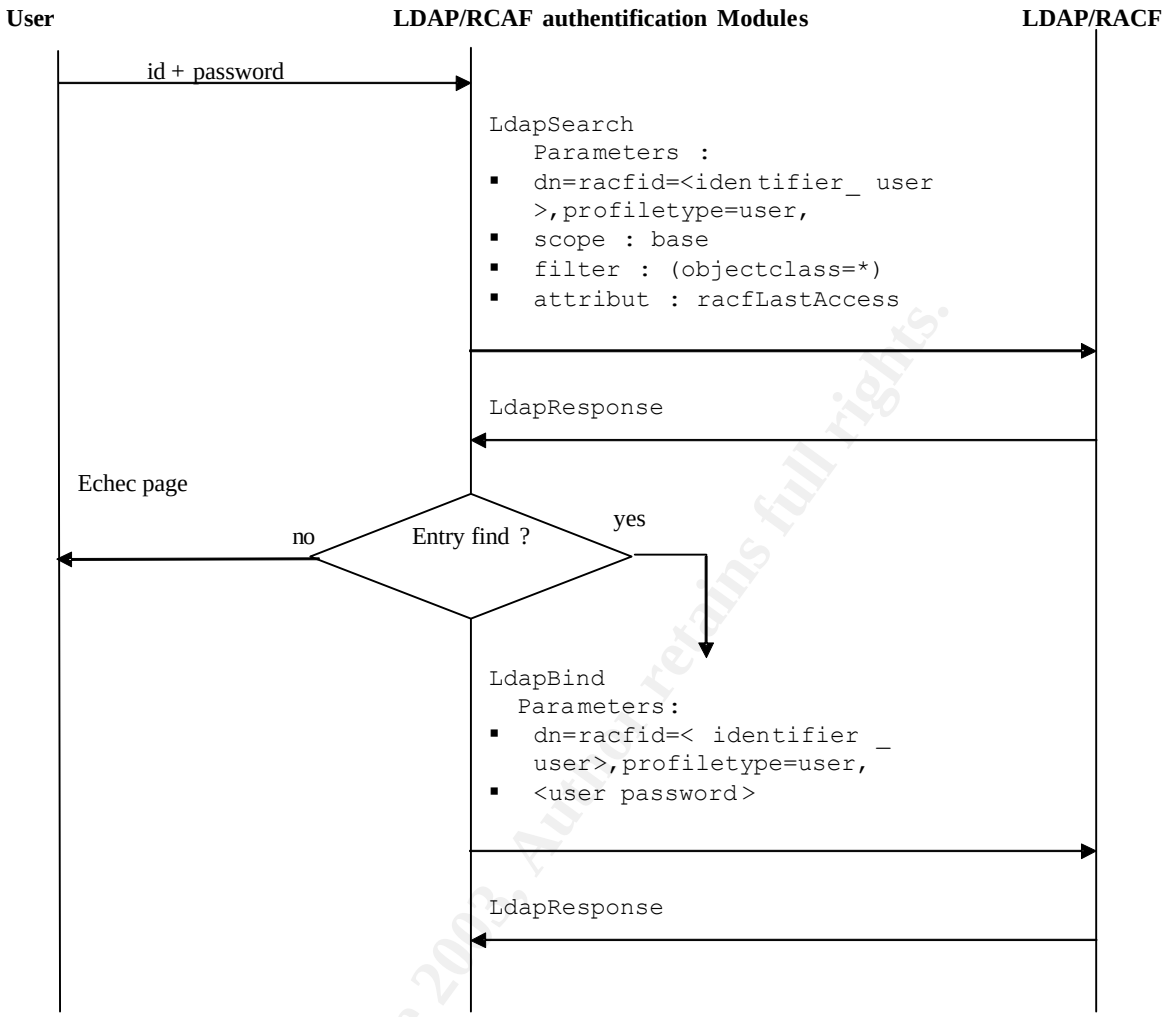
4.2.2 Flows:

Connection flows are as follows:

- Navigator launching
- Browsing a Web portal URL's.
- Connecting into our TAM reverse proxy server,
- Doing a double authentification by CDAS module :
- Doing a strong authentification of working stations by X 509 certificate: checking of the certificate validity (date of validity not expired, certification authority recognized and not belonging to a certificate revocation list),
- Performing a user authentification by HTTP form (login/password) and checking against RACF. This is done according to a password policy. This login form is protected by SSL in order to ensure a confidentiality of exchanges.



Double authentification principles

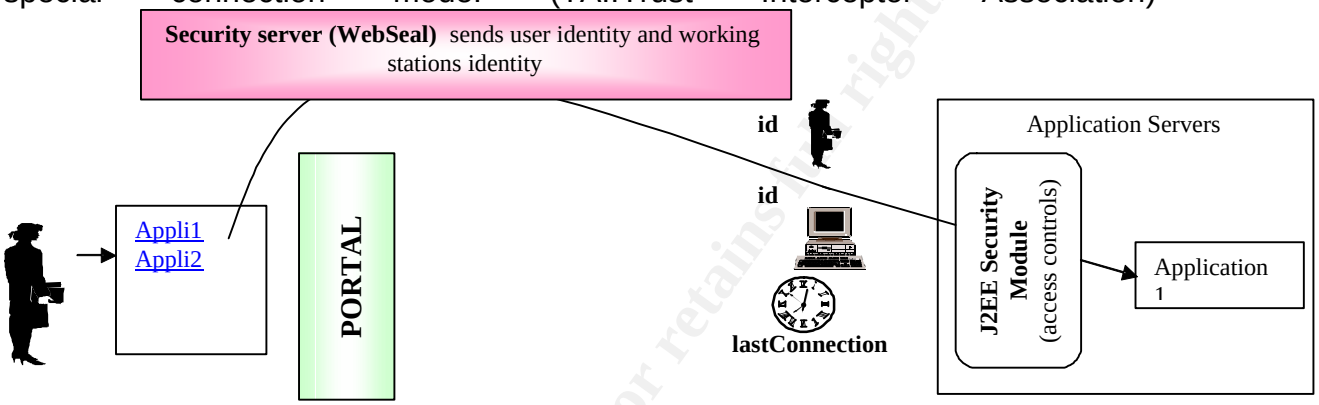


- After authentification, establishment of connection to ta Web portal.
- A security context is propagated into HTTP headers. This one contains following information: user identity, user profile, place-of connection, working station identity, date and hour from the last connection and so on.
- Control rights and authorization of this user according to his profile.
- Seeking in the Ldap database,for information necessary to build a virtual office,
- Choice of an application by this user:
  - Authentification and access control by J2EE,
  - User connection to an application server lodging his selected application.
- Access control to applications or resources via J2EE according to roles protecting these applications or ressources. These controls are carried out by interrogation of a LDAP database.

4.3 WEBSEAL JUNCTION TO MIDDLE END SERVERS

A junction represents a TCP/IP connection between a frontal server and a background Web server.

A connection between WebSeal reverse proxies and middle end servers used a special connection mode. (TAI:Trust Interceptor Association)



4.4 USER CONTEXT

4.4.1 User context propagation

A security context is transmitted by using "Entitlements Services". It is quite simply a question of transmitting into HTTP header a certain number of information related to a user according to such a form "parameter = value".

This method allows good performances by avoiding multiple Ldap readings from each application server.

4.4.2 Security context content

Security context contents two types of data: permanent data and optional ones:

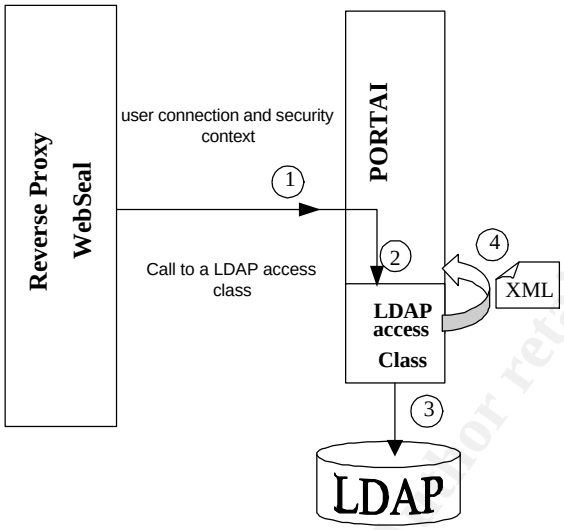
Here you are some example of permanent data:

- User Identifier (⇒ iv_user)
- User DN (⇒ dn_iv_user)
- Name (⇒ Sn)
- First name (⇒ givenname)
- Date and hour from last successful authentification RACF (⇒ racflastaccess) in the form "year.quantieme/hour:minute:seconde". Example: 02.281/10:43:33
- Number of days before password expiration (⇒ delta_password)
- Profile 1 (⇒ profil_1)

- Profile DN 1 (⇒ dn_profil_1)

5 WEB PORTAL ACCESS AND VIRTUAL OFFICE

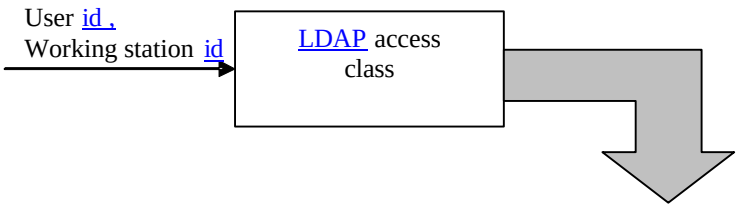
To build a virtual office for each user, a web portal access initially asks a directory LDAP via a LDAP access class. This class uses information from security context generated by WebSeal. This class carries out access LDAP described below and provides a XML flow to the calling class.



Principles of research by using any DN communicated by a security context are as follows:

- By using a connection entity/site DN, LDAP access class seeks some applications available for this entity. Some entries contains particular data such as URL, and other parameters making it possible to build later a virtual office.
- For each applicative entry < DNApplication > previously found, a direct research is done for looking for all applicative roles having a pointer on this particular user's DN. Finally, URLs of these applicative entries are authorized for this user. in accordance with this filtering

LDAP access classes return XML information. For a user prospective, this information is used by the web portal to build some virtual office according to his profile, his connection sites and application roles earned.



.

```
< VIRTUAL_ OFFICE>
   < LAST_CONNECTION_DATE>[date and hour of the last connection]</LAST_CONNECTION_DATE >
   < dn=[DN APPLICATION of the application ] type=intranet|appli of PdT subject=[subject of the application ] >
      < URL>[URL of the application]</URL >
      < TARGET>[Window Targer]</TARGET >
      < PARAMETER name=[Parameter name]>[valor of the parameter]</PARAMETER >
      …
      < PARAMETER name=[Parameter name]>[valor of the parameter]</PARAMETER >
      < GROUP>[ application group]</GROUP >
   </APPLICATION >
  …
</VIRTUAL_OFFICE >
```

Data Format Exchange between a LDAP access class and a calling class

6 ACCESS CONTROL

6.1 PROBLEMS

An information contained in a directory indicating that certain applications are accessible only to certain establishments or accessible solely for certain channels is exploited only for bulding a user's virtual office.

In particular, it is not quite possible to implement access controls in a rigorous way because nothing can prevent a user to directly typing a URL in the "address" field of his navigator. For example, a user with a "working station-without certificate" can directly reach an application which needs a working station authentification.

Controls of a security context transmitted from our reverse proxy to any application servers must prevent a majority of these unauthorized accesses J2EE

applicative roles were chosen to do these controls into each middle end application servers.

The true solution consists in implementing access controls directly in our TAM-reverse proxy but this solution will be implemented in a later version

6.2 WEBSEAL AUTORIZATIONS

WebSeal offers an implementation of authorization services but we choose do not use this possibility for this first version.

6.3 PROFILES AND –RÔLES ACCORDING TO J2EE STANDART

6.3.1 Definitions

- **User profile**: A profile is thus a data related to the user who determines his access rights to a whole of application and possibly, his rights inside an application. A user profile is mainly used on the web portal level.
- **Applicative profile or applicative roles**: An applicative profile or an applicative role has a less range since it is not defined and it is used for protecting a precise application.

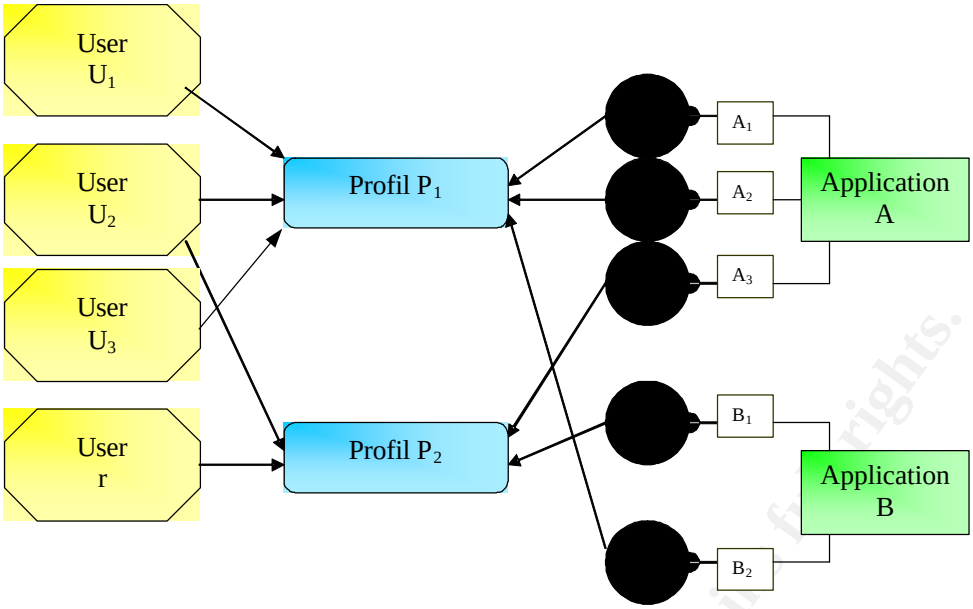
An applicative role can be addressed to a whole of user profiles.

Applicative profiles are implemented into Web/Ejb applications through standards referring to "J2EE".

6.3.2 Relations profils - roles

There is thus a correspondence between trade profiles and J2EE applicative roles. This correspondence is carried out in our LDAP directory.

The diagram below clarifies this correspondence:



7 APPLICATIONS HARDENING: J2EE SECURITY MODEL

Application security must be as much as possible external of its code.

- Security server carries out **authentification** working stations and users.
- Application servers handle **access controls** defined by the concept of applicative profiles.

An implementation of applications according to this model makes it possible to delegate in theory to application servers:

- Access to reference security frames users (directories LDAP, databases…)
- Realization of authentification (it is not the choice of this architecture)
- Realization of access controls to Web resources or EJBs methods according to J2EE role concepts.

7.1 Responsibilities

An application defines a whole of roles and rights associated to these roles:

- ➔ in **a declaratory way** in deployment files
- ➔ In a **programming way** by using JAVA code such as isUserInRole method of javax.servlet.http.HttpServletRequest (for the servlets) or isCallerInRole of javax.ejb.EjbContext (in EJBs).

Directory LDAP contains entries for user types or user groups.

Application servers which are interfaces between applications and LDAP directories link correspondences between entries of LDAP directories (users and/or groups) and applicative roles. It carries out definite access controls in a declaratory way into application deployment files.

7.2 Implementation :

Into applications

Application deployment files allow:

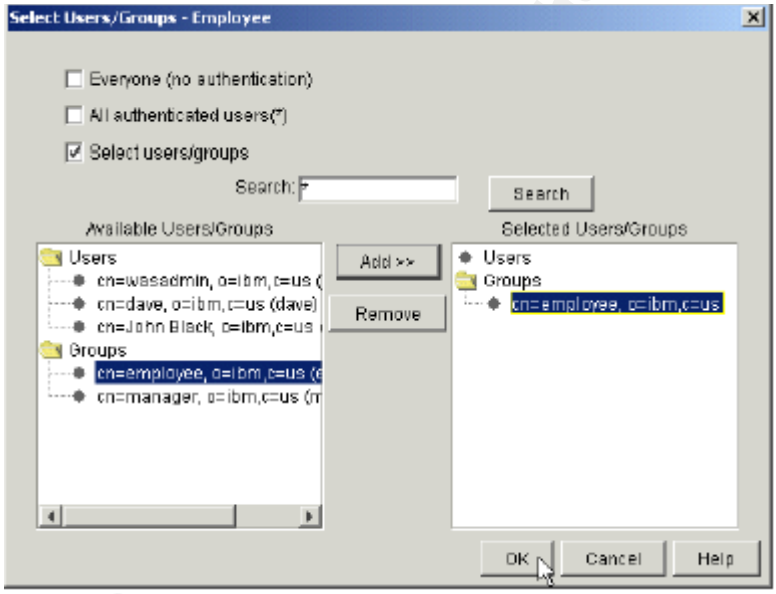
- to declare application roles ,
- to associate roles to WEB (*URL patterns*) resources or EJBs methods.

Into directory LDAP

It is mandatory to define functional groups corresponding to application roles.

Into application servers

Application servers give the possibility to link a whole of user or group entries into LDAP directory to a whole of application roles. All application roles must be associated either with groups within LDAP directory, or with one of the two special groups of the application server: an "All the users" group referring to all users (authenticated or not) and an "authenticated Users" groups referring to only all authenticated users.



7.3 Protecting WEB (URL) resources (security declaratory)

Security constraints specify the way in which a whole of resources (URL + HTTP methods) is protected (resource collection) by specifying roles that users reaching these resources must have (authorization constraint) and constraints of transport (use of SSL or not) to reach these resources.

Extract of WEB application deployment file (web.xml) describing security constraints :

```
<security-constraint id="SecurityConstraint_1">
```

```
        <web-resource-collection id="WebResourceCollection_1">
                <web-resource-name>WebBank Transfer</web-resource-name>
                <url-pattern>/</url-pattern>
                <url-pattern>/webbank.html</url-pattern>
                <url-pattern>/TransferServlet</url-pattern>
                <http-method>GET</http-method>
                <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint id="AuthConstraint_1">
                <description>All role for TransferServlet:+:</description>
                <role-name>AllAuthenticated</role-name>
        </auth-constraint>
        <user-data-constraint id="UserDataConstraint_1">
                <transport-guarantee>NONE</transport-guarantee>
        </user-data-constraint>
</security-constraint>
```

7.4 Need to making secure EJB

EJBs can be divided between several applications; this is why it is significant that access to these components should be controlled in order to avoid an improper use by one of these applications.

7.4.1 Roles declaration of the and protection of methods in deployment files (ejb-jar.xml)

Declaration of a role name used by an implementation code (method isCallerInRole):

…

```
...
<enterprise-beans>
  ...
  <entity>
    <ejb-name>AardvarkPayroll</ejb-name>
    <ejb-class>com.aardvark.payroll.PayrollBean</ejb-class>
    ...
    <security-role-ref>
      <role-name>payroll</role-name>
```

```
        <role-link>payroll-department</role-link>

      </security-role-ref>
    ...
  </entity>
   ...
</enterprise-beans>
...
```

…

A role name is specific to an EJB and its declaration into a deployment file is a duty of EJB developers.

Application roles Declaration:

```
<assembly-descriptor>
   <security-role>
      <description>
         This role includes the employees of the payroll
         department. The role is allowed to view and
         update the payroll entry for any employee.
      </description>
      <role-name>payroll-department</role-name>
   </security-role>
   ...
</assembly-descriptor>
```

How we can define permissions for invocation method

…

```
...
<method-permission>
   <role-name>employee</role-name>
   <method>
      <ejb-name>EmployeeService</ejb-name>
      <method-name>*</method-name>
   </method>
```

```
</method-permission>
<method-permission>
  <role-name>employee</role-name>
  <method>
    <ejb-name>AardvarkPayroll</ejb-name>
    <method-name>findByPrimaryKey</method-name>
  </method>
  <method>
    <ejb-name>AardvarkPayroll</ejb-name>
    <method-name>getEmployeeInfo</method-name>
  </method>
  <method>
    <ejb-name>AardvarkPayroll</ejb-name>
```

…

Interfaces methods ("remote interface" and "home interface") of an EJB can be protected in a declaratory way into an application deployment file.

The provided examples are extracted from the specification "Enterprise JavaBeans™ Specification, v1.1" available to the URL following: http://java.sun.com/products/ejb/docs.html.

7.4.2 Role controls and J2EE APIs

IsUserInRole of javax.servlet.http.HttpServletRequest (for servlets) or isCallerInRole of javax.ejb.EjbContext (for EJBs) determine if a user possesses a role.

These methods can be used for:

- Building a menu,
- Implementing level 2 security.

Building menus

Let's s say an application whose access controls are defined by 3 roles (R1, R2 and R3).

A page-of menu of this application indicates the functionalities accessible to a user according to his roles. The following table details functionalities of a menu accessible to each role.

FUNCTIONALITIES	R1	R2	R3

FUNCTIONALITIES	R1	R2	R3
Application 1	Y	N	Y
Application 2	N	Y	Y
Application 3	N	N	Y

According to the roles of the user, 4 menus are possible:

- Menu 1 (if a user has the R3 role):

Application 1
Application 2
Application 3

- Menu 2 (if a user has roles R1 and R2 and not R3 role):

Application 1
Application 2

- Menu 3 (if a user has a R1 role and not a R2 role, nor a R3 role):

Application 1

- Menu 4 (if a user has a R2 role and not a R1 role, nor a R3 role):

Application 1

Example of implementation in a Servlet:

```
public static String R1 = "R1";
public static String R2 = "R2";
public static String R3 = "R3";
…

Protected void displayMenu (HttpServletRequest req, HttpServletResonse resp)
  …
  If (req.isUserInRole(R1) || req.isUserInRole (R3))
    //to show a link towards the functionality "Application 1"
     …
```

If (req.isUserInRole(R2) || req.isUserInRole (R3))

//to **show a link** towards the functionality "Application 2"

…

If (show a link towards the functionality "Application 3"

…

8 FLOW INTEGRITY AND CONFIDENTIALITY

8.1 PERIMETER

Data stored on servers, databases and directories are not encrypted except for passwords. Data protection is ensured by automatic checks (access control lists – ACL) configured on various servers. Tripwire and different security operating system utilities are also used to.

8.2 PRINCIPLES AND IMPLEMENTATION

Flow Integrity and confidentiality are mainly based on use of standard such as SSL. Some technical security specifications will precisely determine SSL parameters (algorithms, keys lengths) as indicated by [3]

Authentification phases and some sensitive must be protected by using confidentiality and integrity tools. For example security administrative flows, personal information and accounting transactions are well known as sensitive information.

The different user authentification phases must imperatively be encrypted and sealed in order to protect password transport.

If applicative exchanges are considered to be sensitive, then they must be sealed, encrypted and sometimes sealed too.

Administrative security exchanges (LDAP and RACF flows) must be encrypted and sealed.

9 HARDWARE AND SOFTWARE INTEGRITY

Integrity is ensured by physical and logical protection measures.

Physical protection relates on buildings and physical monitoring of people. This is outside of our scope.

Logical integrity is guaranteed by several types of measurements of precautions:

- UNIX Security controls based on an integration of an integrity checking agent and practical application of CERT.
- Usage of European agencies recommendations for Z/OS
- Respect of correct processes regarding patches and updates
- Real defense in depth architecture regarding network
- Differentiation between development, tests and exploitation environments:
- An authenticated user in a test environment should not be able to reach operating data;
- Security measures for testing means must be differentiated from those of production. In particular, the certificate management servers for testing purposes should not be able to issue cryptographic certificates which would be valid and thus usable in production.

10 AVAILABILITY

Security operations and architecture should have an availability level equal or higher than those set up for applicative
Authentification processes must have a high level of availability. In their absence, access to any resources is prohibited.

This kind of required availability level is reached by using an active redundancy for servers. Some special attention is focused on TAM servers regarding its key place in our architecture

11 LOGGING AND WARNING TOOLS

Rules for logging and audit are describes as follows:

- An evidence of imputability is required from beginning to end. For that, user identity must be propagated on all the chain of connection in order to ensure a logging of operations.
- Significant security events and other specific events related to any requirements of applicative must be logged. They should collect user identifiers that are at the origin of these events.

Here you are different examples.

- Tivoli Access Manager: Registering of administrator identities on duties during inscription, radiation, modification of any security attributes, any privileges or any policy rules on any bases and directories managed by TAM.
- PKI: Recording of all events - and their authors - related to cycle of life certificates: emission, revocation, renewal…
- WebSeal: Recording of all significant events of operation: rejected, try-to access to an URL protected etc
- WebSphere: Recording of all connection established, all attempts for accessing to some protected resources (objects or methods) incompatible with rules, the roles and groups
- LDAP and RACF. Recording of all events - and their authors - related to security administrative tasks. Each object of theses bases will have to comprise nonmodifiable attributes in order to log any author having created or modified this object. Date and time of these should be logged.

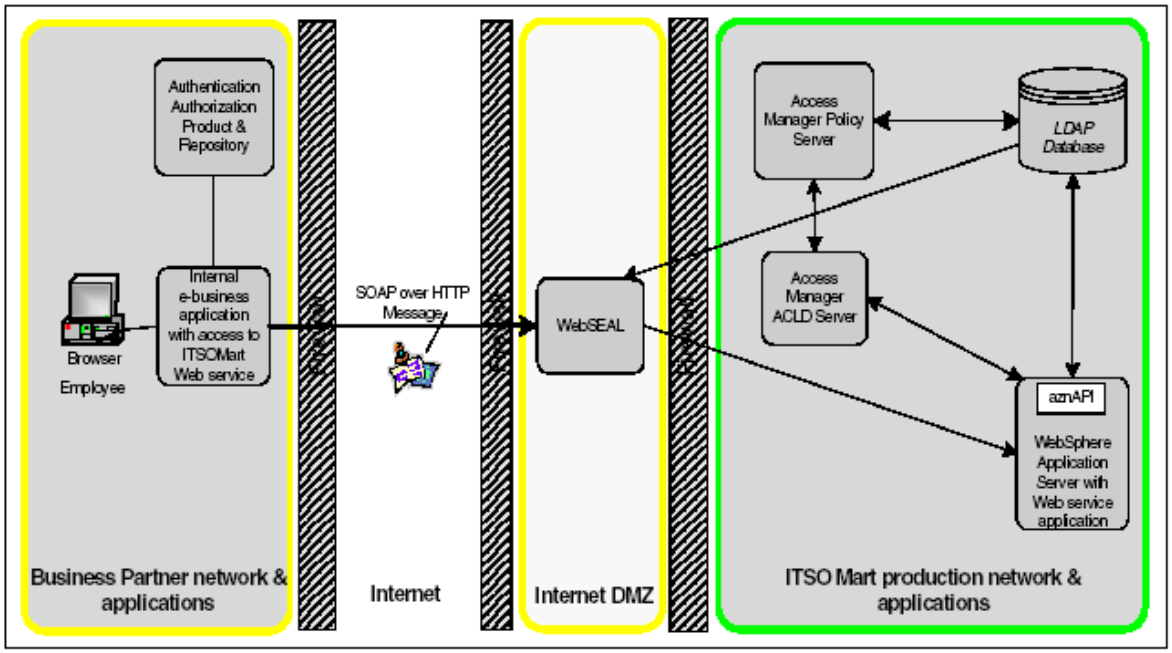
- All logging and warning journals should be exploited by adequate tools.

12 FUTURE

This project was implemented and deployed as described without major problems.

However new evolutions are being studied and will cover the following points:

- Study and deployment of a true PKI in order to allow strong authentifications of users and servers. This study is not only technical (choice of best technology applicable to our environment) so we will focus first on reengineering of our business processes. This next challenge is more organizational than technical.
- Study and implementation of an equivalent architecture for banking on Internet and Extranet).
- Integration of WebServices in a future version. [6]



Possible integration of WebServices (source [6])

13 CONCLUSIONS

This project demonstrates that it is possible to implement a concept of SSO which was often described as theoretical without real practical application. It proved the feasibility of such an implementation.

The complexity of our organization gave limits to this study. However first experience feedback shows that this concept is finally available and adapted to complex organizations on a great scale

An implementation of J2EE security frameworks, good practices in development and a deployment of a secure centralized database in correlation with Tivoli Access Manager Server pushing security policies into a distributed component (Webseal), show us that this technology is mature.

However an appropriate use of standards for interoperability purposes is mandatory.

REFERENCES AND BIBKIOGRAPHY

Enterprise JavaBeans™ Specification, v1.1: http://java.sun.com/products/ejb/docs.html. [1]

Information Security Management Handbook (4 HT Edition) (Tripton Krause) Edition: Auerbach [2]

Hack Proofing Your E-Trade Sit (Ryan Russell, Teri Bidwell, Oliver Steudler, Robin Walshaw, L Brent Houston) Edition: Syngress [3]

Web Single Sign One Meets Business Reality Tim Mather: http://www.sans.org/rr/authentic/reality.php [4]

Enterprise Bussiness Portals with IBM Tivoli Access Manager http://www.redbooks.ibm.com/redbooks/SG246556.html [5]

Enterprise Bussiness Portals II with IBM Tivoli Access Manager] http://www.redbooks.ibm.com/redbooks/SG246885.html [6]

Another document not referred in this paper:

Configuration of TAM: http://grumpy.e-enterprise.purdue.edu/wps/doc/en/InfoCenter/wpf/conf_pdr.html

WebSeal installation guide (version 3.9) Amweb39_install.pdf

WebSeal administration guide (version 3.9) Amweb39_install.pdf

Tivoli Access Manager Administration guide (version 3.9) Am39_admin.pdf

Glossary:

J2EE	Java 2 Enterprise Edition: standard Java for the development of applications.
TAM	Tivoli Access Manager
SSO	Single Sign One: single authentification
WAS	WebSphere Server Application: a application server from IBM
DN	Distinguished Name
CDAS	Cross-country race Domain Authentication Service: modulate integrated into WebSeal which manages the phase of authentification of Tivoli Access Manager
RACF	Resource Access Facility Control
CRL	Certificate Revocation List