CENTRALIZED AUTHENTICATION SERVICES

M.S.Vinmathi¹, B.Mohan², Mr. Prabharakar.³
¹Asst.Professor, ²P.G. Scholar, ³Faculty
¹²³ Dept. of M.C.A., Panimalar Engineering College
Anna Univeristy, Chennai

Abstract -- Centralized Authentication Service (CAS) is a single sign-on protocol for the web. Its purpose is to permit a user to log into multiple applications simultaneously and automatically. It also allows web applications to authenticate users without gaining access to a user’s security credentials, such as a password. Single sign-on allows a user to enter his or her name and password once and gain access to multiple applications or systems. Utilizing this authentication process eliminates the need for multiple prompts when the user switches from one application to the next.

Web single sign-on works strictly with applications accessed with a web browser. The request to access a web resource is intercepted either by a component in the web server, or by the application itself. Unauthenticated users are diverted to an authentication service and returned only after a successful authentication.

Related Glossary Terms: CAS Centralized Authentication Services.

INTRODUCTION

Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. In private and public computer networks (including the Internet), authentication is commonly done through the use of logon passwords. Knowledge of the password is assumed to guarantee that the user is authentic. Each user registers initially (or is registered by someone else), using an assigned or self-declared password. On each subsequent use, the user must know and use the previously declared password. The weakness in this system for transactions that are significant (such as the exchange of money) is that passwords can often be stolen, accidentally revealed, or forgotten.

For this reason, Internet business and many other transactions require a more stringent authentication process. The use of digital certificates issued and verified by a Certificate Authority (CA) as part of a public key infrastructure is considered likely to become the standard way to perform authentication on the Internet. Logically, authentication precedes authorization (although they may often seem to be combined).

AUTHENTICATION

Authentication is the process of obtaining identification credentials such as name and password from a user and validating those credentials against some authority. If the credentials are valid, the entity that submitted the credentials is considered an authenticated identity. Once an identity has been authenticated, the authorization process determines whether that identity has access to a given resource.

The precondition for access control is to make sure that the person or program requesting access is identified without doubt. This process is called authentication. In its most common form, authentication employs a user identification string (user ID) and a user secret (password) to establish certainty about the requestor. The standard ways to supply the user ID and password to the server are HTTP basic authentication (a Web browser mechanism using a standard login dialog), or HTTP form-based authentication (a mechanism in which the server sends a customized authentication form that is presented to the user). Other authentication mechanisms include SSL/TLS client authentication based on digital signatures and certificates (including secure storage on smart cards), hardware-based one-time passwords such as the RSA SecureID token, and various biometric mechanisms such as fingerprint verification, iris scanning, or speaker verification. The WebSphere Portal authentication subsystem delegates the task of authentication to underlying mechanisms of WebSphere Application Server. It provides support for the following authentication configurations:

- Use of native WebSphere Application Server authentication. A custom login form posts the user’s authentication data to a servlet which requests WebSphere Application Server Security to validate the user’s authentication data. Alternatively, you can set up WebSphere Application Server to accept and validate SSL client certificates. This setup exploits the fact that WebSphere Portal is well integrated with WebSphere Application Server and that the portal can be configured as a secure Web application.
- Use of an authentication proxy. WebSphere Application Server provides an interface that allows it to establish trust to authentication proxies; this interface is called the Trust Association Interceptor interface.

USING WEBSHERE APPLICATION SERVER SECURITY

To use WebSphere Application Server Security, the portal is configured as a secure Web application. When WebSphere Application Server receives a request for this application (the portal), its security component requests the authentication credentials from the client. Depending on which authentication method you configured, one of the following occurs,
**An HTTPS client certificate request is sent to the browser,**

**The client is redirected to an authentication form that prompts the user to provide the credentials for authentication.**

In the latter case (which is the portal default configuration), the form posts the credentials to WebSphere Portal’s custom authentication servlet which obtains the posted credentials. The servlet then logs the user in to WebSphere Application Server’s security context using the JAAS (Java Authentication and Authorization Services) Framework, as described in JAAS Login.

![CAS Authentication Mechanism](Image)

**Fig 1: CAS Authentication Mechanism**

In the case of HTTPS client certificate authentication, WebSphere Application Server receives the authentication credentials directly. WebSphere Application Server authenticates the user by checking the provided credentials against the user registry, which can either be an LDAP directory or a custom user registry. Figure 1 shows the detailed flow of control when a request for a portal destination is not yet authenticated, and needs to be authenticated by WebSphere Application Server Security. In this example, the configuration was set up to use HTTP form based authentication. The post-login page can be configured as part of the post-login redirection policy.

![Authentication flow with WebSphere Portal configured to use WebSphere Application Server Security and HTTP form-based authentication](Image)

**Fig 2: Authentication flow with WebSphere Portal configured to use WebSphere Application Server Security and HTTP form-based authentication**

CAS also supports proxy authentication. Proxy authentication is useful for middle-tier applications that need to connect to another application on behalf of the user. For example, you visit a portal that requires you to be logged in to CAS. This portal shows information from Blackboard, which uses CAS. Proxy authentication can be used so the portal can obtain information from Blackboard without the user explicitly logging in to Blackboard. Proxy authentication can be very confusing and can also lead to security concerns if not implemented properly. A good source on proxy authentication is the official documentation on proxy authentication, the CAS Architecture, and the CAS protocol.

**Request in a session which is already authenticated**

After successful authentication (through either WebSphere Application Server Security or an authentication proxy with a TAI) the user is logged in. Then, the user's authenticated portal session starts and an SSO token is issued to the client as a cookie, along with the HTTP session cookie (usually named JSESSIONID). The default implementation used for the SSO cookie is the Lightweight Third Party Authentication (LTPA) scheme supported by the WebSphere components. As proof of authentication, user information is signed and encrypted into an LTPA token, which can be verified by all servers that are part of this LTPA SSO domain. An LTPA token contains the user ID and has an expiry date and time, at which time it expires (even when continuously used). It is possible to use custom SSO tokens to obtain trust between calls from the client.

![Flow for a request that is already authenticated](Image)

**Fig 3: Flow for a request that is already authenticated**

![USER REGISTRY](Image)

**USER REGISTRY**
The user registry is a repository that holds information about registered users and groups. WebSphere Application Server and WebSphere Portal support an internal WebSphere Portal database, an LDAP directory, or a custom registry as a user registry. WebSphere Portal shares the authentication registry with WebSphere Application Server and can also (optionally) have a separate look-aside database for user profile data and preferences. Some profile information can also be stored in the same physical store as the authentication registry. For example, an LDAP directory typically contains much more information about each user than just the name and password.

In WebSphere Portal, the information about users is centralized in the WebSphere Member Manager (WMM) component. You can configure this WebSphere Portal component for different layouts of data in the user registry and its look-aside database. Also, WebSphere Portal can work with a read-only user registry, in which case, all portal user data that needs to be updated is stored in the internal look-aside database. The WebSphere Member Manager component is also responsible for determining group membership. This information is used by WebSphere Portal’s access control component and administration functions. The lookup is able to evaluate nested groups. Because this lookup has a potential impact on performance, a configurable option lets you stop group membership lookup at the first level.

PORTAL LOGIN

After authentication to WebSphere Application Server, the portal login is performed. The portal login accepts the authenticated user ID string from the WebSphere Application Server security context and initializes the portal’s HTTP session with corresponding user data:

- The user profile data is populated from the user registry.
- The user session is resumed from the saved state (if this option was chosen).
- The browser is redirected to post-login target page.

In addition, you can configure your portal to perform a second portal-specific JAAS login to populate the portal user object with additional credentials.

When the user terminates the session, he or she can either explicitly perform a logout action, or the session can time out.

LOGOUT

The logout can be initiated by a URI that is explicitly selected when the user clicks on the logout button (which has been added by the aggregation engine to every page banner). Alternatively, the logout can be initiated implicitly through a session timeout that occurs after a specified time of inactivity.

The portal logout performs the following actions:

1. **Suspend User Session.** The user’s portal session (the portal’s navigational state) is persisted. The HTTP session is invalidated.
2. **Portlet user logout.** The portlets are notified of the event “user logged out” to give them the opportunity to finalize (trans-)actions that need to be terminated.

The following steps are performed only when the logout is initiated by an explicit user action.

They are not performed when the logout is a consequence of a timeout.

3. **WebSphere Application Server logout.** The user’s credential token is marked as invalid, and a respective cookie invalidation command is added to the response.
4. **Browser redirect (302).** The browser is redirected to render the post-logout target.

In the case of an authentication proxy, WebSphere Portal needs to be set up to redirect after logout to the authentication proxy’s logout page. This way the authentication proxy is also notified of the logout. If a user does not logout explicitly, the proxy cannot be notified of the portal session timeout and the proxy session will time out eventually as well.

SINGLE SIGN-ON (SSO)

WebSphere Portal is often used to integrate other enterprise information systems, and to present them through the portal user interface. The recommended approach is that those systems do not relinquish control of their application security, even if they are accessed through WebSphere Portal. Therefore, the back-end systems continue to do their own authentication and authorization. However, most implementations want to avoid requiring the user to repeatedly authenticate. This is where SSO comes into play. The goal of SSO is to provide a secure method of authenticating a user one time within an environment, and to use that single authentication (for the duration of the session) as a basis for access to other applications, systems, and networks. In the context of WebSphere Portal, there are two SSO realms,

- From the client to portal and other Web applications
- From the portal to the back-end applications.

Those two realms are indicated in by loops.

Fig 4: Portal single sign-on realms

With single sign-on from the client to the Web applications and the portal, a client logs into one Web application (once), and can then access all other Web applications that are part of...
the same SSO realm, without receiving a second authentication challenge. It makes no difference whether WebSphere Portal, or one of the other trusted Web applications, is the authenticating Web application. Single sign-on from the portal to the back-end applications enables a portal client to log into the portal once, and then to access a number of back-end applications through respective portlets, without having to authenticate at each of these applications.

SECURING CLIENT –TO-PORTAL COMMUNICATION

Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS) are protocols that leverage a variety of cryptographic algorithms to implement security, for example authentication with certificates, session key exchange algorithms, encryption, and integrity checks. They are common protocols that are often used to provide privacy and trust between communicating applications such as Web clients and Web servers. When HTTP is used over such a secure connection, this is known as HTTPS. SSL and TLS provide connection security with three basic properties:
1. **Confidentiality** - Encryption is used after an initial handshake to define a secret key. Symmetric cryptography is used for data encryption (for example, DES, RC4)
2. **Authentication** - The peer's identity can be authenticated using public key cryptography (for example, RSA, DSS).
3. **Integrity** - Message transport includes a message integrity check using a keyed Message Authentication Code (MAC). Secure hash functions (for example, SHA, MD5) are used for MAC computations. While TLS is the standard implementation, SSL still is more widely distributed. In this description, we refer to SSL, but also imply TLS. WebSphere Portal is tested and supported to run with the FIPS 140-2 compliant TLS providers available in WebSphere Application Server.

**IPSec** is a technology that can be used as an alternative to SSL. It works on a lower layer of the OSI reference model than SSL and is usually configured and handled on the operating system level. Although the operating systems of most clients support IPSec today, only a few have it already configured. Therefore, we do not recommend using it for the communication between clients and the server complex. When the communication between servers needs to be protected however, it is a question of design whether to use SSL or IPSec. Both technologies offer advantages: IPSec is on the network level and is, therefore, transparent to applications; they do not have to do anything to support it. In addition, it might perform faster if the OS makes use of available crypto hardware. SSL, on the other hand, is widely supported and administrators have gained a lot of experience and confidence with this technology over the years.

WebSphere Portal can be configured to support SSL. For example, it can generate respective URLs if needed; you can specify the protocol as well as the port. The main questions with using transport or network layer security are where to terminate the protected connections and which communication to protect. If a higher level of security is desired, it is necessary not only to protect the communication across unprotected networks (such as the Internet), but also the communication within the corporate networks. The threats of human errors that might open up holes in the network protection as well as the threats of internal attackers (such as disgruntled or dishonest employees) are not to be underestimated. In fact it is estimated that in about half of the cases, these are the causes of security incidents.

Fig 5: Communication security for portal installations within a protected network

The second of our deployment scenarios separates the Web server from the portal server with firewalls between the client and Web server, and between the Web server and portal server.

Fig 6: Communication security for a portal installation in a trusted intranet

The third of the deployment scenarios (Figure 4) places a reverse proxy for authentication in the DMZ and, behind another firewall, a cluster of portal servers that are fed by a load balancer. A possible secure communication implementation for this case could use SSL from client to reverse proxy, and IPSec for the remaining path between the servers: one IPSec connection from authentication proxy to the load balancer, and another one from load balancer to the portal cluster as shown in Figure 6.

Fig 7: Communication security for a straightforward Internet portal cluster

In a fully clustered portal environment, with full transport layer security, the client’s SSL connection is tunnelled through the first load balancer and terminated at the authentication proxy cluster. The tunneling can be done because the first load balancer does not need to read the request’s content for its load balancing strategy. That is only needed for more sophisticated strategies such as session affinity, for example, when all requests by the same user are always routed to the
same cluster machine. The second load balancer will usually use such a strategy; therefore, the next SSL connection needs to be terminated at this server. Finally, there is a third SSL connection between load balancer and portal cluster. For this scenario, we have shown how to connect to an LDAP directory. The communication between application server and LDAP server is protected as well because the LDAP queries and responses might contain confidential data. You can configure WebSphere Application Server’s security component to use LDAPS (LDAP over SSL)

Restricting the protection to sensitive communication only Using SSL is not cheap; it costs a considerable amount of computing power. The most expensive SSL operation is the initial handshake, but the symmetric encryption of the bulk data produces additional load. Therefore, limit the SSL protection to the necessary minimum. WebSphere Portal can switch between SSL and non-SSL connections during a session. You can set up SSL for the portal with these configurations:

- For the user login interaction only,
- For login and the part of the portal that is restricted to authenticated users,
- For the whole portal,

The final decision to whether a portal should protect none, all, or only parts of its communication with SSL should be based on the sensitivity of the data.

CONCLUSION

The project is identified by the merits of the system offered to the user. The merits of this project are as follows,

- It’s a web-enabled project.
- This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information through so much simplicity.
- The user is mainly more concerned about the validity of the data, whatever he is entering. There are checks on every stages of any new creation, data entry or updation so that the user cannot enter the invalid data, which can create problems at later date.
- Sometimes the user finds in the later stages of using project that he needs to update some of the information that he entered earlier. There are options for him by which he can update the records. Moreover there is restriction for his that he cannot change the primary data field. This keeps the validity of the data to longer extent.
- User is provided the option of monitoring the records he entered earlier. He can see the desired records with the variety of options provided by him.
- From every part of the project the user is provided with the links through framing so that he can go from one option of the project to other as per the requirement. This is bound to be simple and very friendly as per the user is concerned. That is, we can sat that the project is user friendly which is one of the primary concerns of any good project.
- Data storage and retrieval will become faster and easier to maintain because data is stored in a systematic manner and in a single database.
- Decision making process would be greatly enhanced because of faster processing of information since data collection from information available on computer takes much less time then manual system.
- Allocating of sample results becomes much faster because at a time the user can see the records of last years.
- Easier and faster data transfer through latest technology associated with the computer and communication.
- Through these features it will increase the efficiency, accuracy and transparency.

REFERENCES