WebSphere® software

Integrating WebSphere Portal software with your security infrastructure.

By Ingo Schuster, Frank Seliger,
Dieter Bühler and Thomas Schaeck
IBM Software Group

October 2003

Contents

Executive overview

Portals provide personalized access to information, applications, processes and people from a centralized point. You can authenticate users and control access to various kinds of information and applications, based on preset security definitions. Allow public access to less-sensitive information. And guard classified information – such as business-critical enterprise or government data – more carefully.

To accommodate different security requirements, portal servers must integrate with various security infrastructure components – such as authentication, authorization and single sign-on control– so you can choose the combination that best matches your security needs. For example, authentication might be as simple as requiring users to provide a correct user name and password. Or, your customers could use a “smart card” – for example, a bank card with a chip that safely stores a private key and certificate – to provide authentication. The smart card enables Secure Sockets Layer (SSL) or Transport Layer Security (TLS) client authentication to establish an authenticated and safe connection between the client and the portal.

Through a modular architecture, IBM WebSphere® Portal for Multiplatforms, Version 5.0 allows the integration of different authentication proxies, authorization systems and credential vault implementations. WebSphere Portal software is designed to work with IBM WebSphere Application Server security [1] features and IBM Tivoli® Access Manager, as well as with third-party security products, so that you can build a highly protected system that fits your individual infrastructure.

Introduction

Using WebSphere Portal, Version 5.0, you can establish protected access to portal resources like page groups, pages, portlets and documents. WebSphere Portal offers different ways to perform user authentication and authorization, and it provides support for single sign-on. Figure 1 shows the key components, from left to right, that are invoked when WebSphere Portal software handles requests for portal pages. Although some portal pages can be accessed without prior authentication (anonymously), all requests for personal pages or group pages must pass through an authentication component. If a user is authenticated successfully, the incoming request is analyzed and routed to the appropriate component.

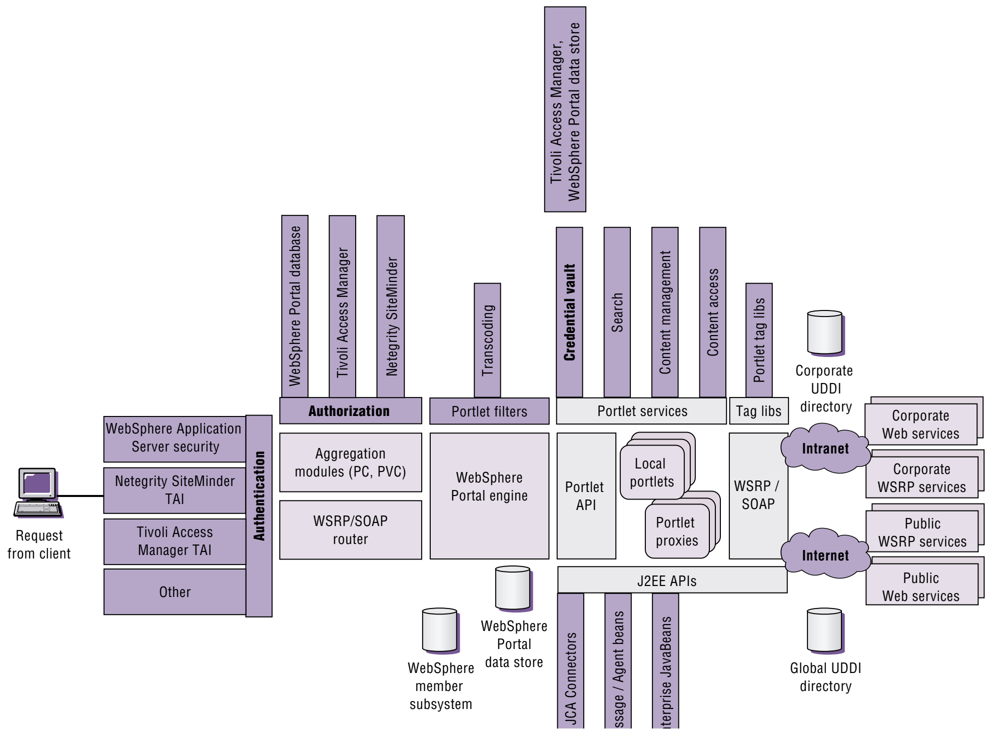

Figure 1. Subsystems of WebSphere Portal, Version 5.0

HTTP requests from user agents seeking a portal page are routed to the portal servlet, which acts as the central access point to all portal pages. Simple Object Access Protocol (SOAP) requests from other portals intended to invoke one of the local portlets are routed to the Web Services for Remote Portal (WSRP) or SOAP router. In either case, authorization is required. The component receiving the request calls the authorization component to determine whether the requested page or the portlet to be invoked can be accessed. To guarantee consistency of access rights, authorization information to most portal resources is maintained in the portal. However, for some resources access control may be externalized to systems such as Tivoli Access Manager or Netegrity SiteMinder.

When a request is routed to the portal servlet and access to a page is authorized, the portal servlet obtains information about the portlets[2] that are referenced from the page. Next, the portal servlet calls the authorization component to determine the subset that the current user may see, according to current access rights, and then displays the subset. Next, the servlet selects the appropriate aggregation module – based on the user-agent information from the request –

and calls that module, passing the information about the page and the portlets to be displayed. The invoked aggregation module renders the page and the portlets in it by calling the referenced portlets. The module uses the portlet application programming interface (API) to communicate additional user profile information and portlet instance data from the WebSphere Portal data store to the portlet.

Portlets are special servlets that—unlike other servlets—are designed to participate in the portal's event model and exploit the portal infrastructure. As a result, a portlet invoked by the portal can, in turn, invoke any Java™ 2 Platform, Enterprise Edition (J2EE) function such as J2EE Connector Architecture (JCA) Connectors, Enterprise JavaBeans (EJB) components, message beans and Java Database Connectivity (JDBC) components. A portlet may also call Web services and HTTP uniform resource locators (URLs) in the same way it calls other servlets. It can also invoke portal-specific portlet services and tag libraries designed to access the portal infrastructure from JavaServer Pages (JSP) components.

Portlets are often used to integrate back-end applications with portals, providing a Web-based user interface (UI) through the portal. When used this way, having single sign-on capability is important because after logging into the portal, users don't want to enter an additional password for each application portlet for the respective back-end systems. To enable single sign-on, WebSphere Portal provides a credential vault portlet service. With this service, credentials are available to portlets to forward to back-end systems where users are then transparently authenticated. The credential vault service is backed by a credential vault, which can either be the portal's built-in vault or a Tivoli Access Manager vault.

Typical portal scenarios

When you build a portal system on WebSphere Portal, you have a variety of possible deployment options—including simple firewall protection and Internet portal-based deployment—that provides different levels of performance, scalability, availability and security.

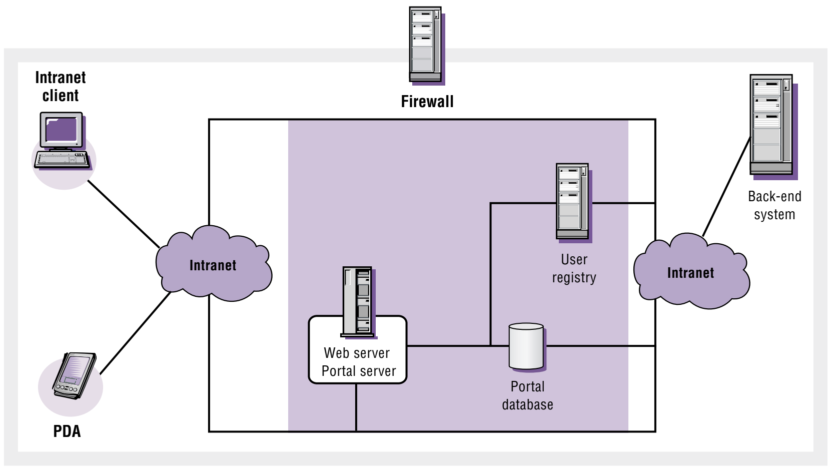

Figure 2. Deployment of a portal within a protected network

Portals in protected networks

You can use a simple portal setup if your portal and its clients reside in your protected network, assuming no internal attacks are expected. Because attacks can be launched from your intranet, this setup, however common, is not the best practice. The portal server, the database server and the user registry and clients are all in the same network, protected by a firewall so they are not visible to the Internet (see Figure 2). The Web server and the application server (with the portal server code) are located on the same physical machine. Clients can be connected to the portal server directly because they are in the same network. The portal uses a separate database server to store data. All user information is in a user registry on a third machine. And the same network includes any back-end systems the portal applications might access.

Internet portals

When you need your portal to service Internet-based clients, simple firewall protection isn't enough. At least one node of the portal system must be visible from the Internet. Typically, two or more firewalls are used, but there are a number of variations you can use.

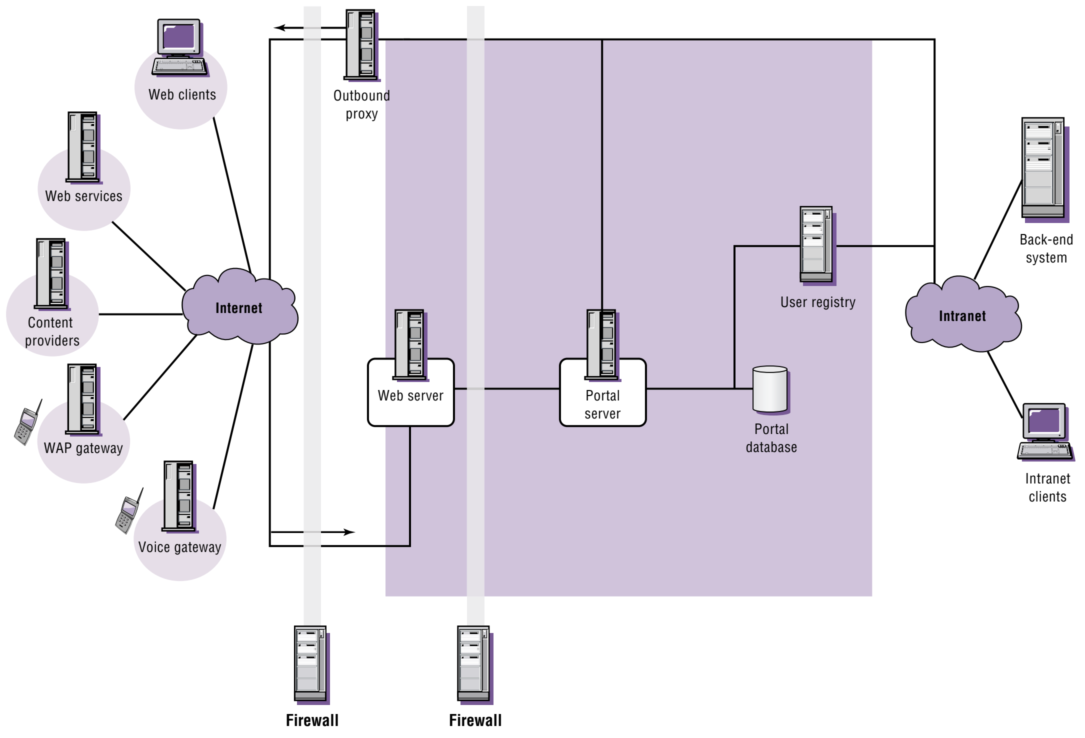

Figure 3. Example of a portal configuration serving to clients in the Internet

The first configuration is appropriate for portals that use WebSphere Application Server security features. If the portal directly handles authentication and access control for its resources, the machine running the Web server must be visible to the Internet (see Figure 3) because the Web server is the first node accessible to external clients. This means that the Web server and the application server with the portal server code must be located on different machines. The Web server is located in a demilitarized zone (DMZ) behind an outer firewall. And HTTP requests to the Web server are the only incoming traffic.

A second firewall provides additional protection for the inner network and the back-end systems by only allowing traffic from the Web server to the portal server. To penetrate the second firewall, an attacker must successfully seize control of the Web server. To allow the portal server to access and aggregate external content, an outbound proxy in the DMZ forwards requests from the intranet and returns the responses from the Internet. You can use additional firewalls to segment your intranet.

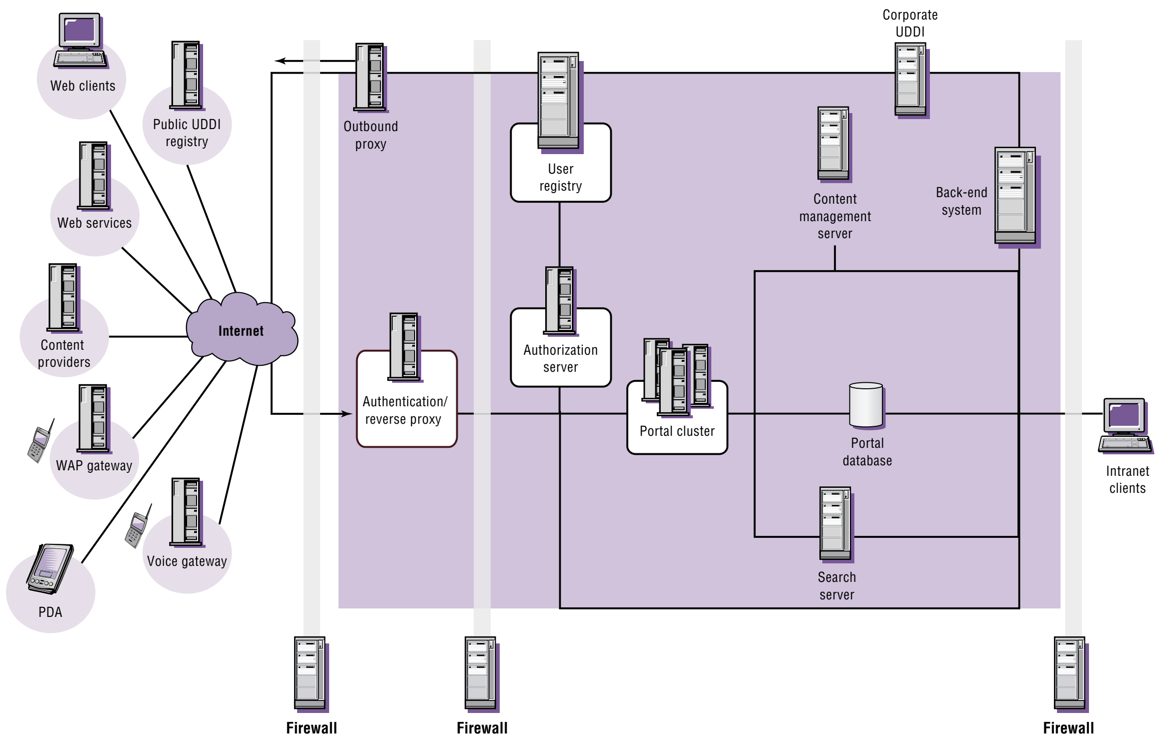

Figure 4. A portal deployment with separate authentication component

The second configuration option is ideal for portals that use a separate authentication proxy. To authenticate, WebSphere Portal can use its own authentication mechanism or rely on a separate authentication proxy, as shown in Figure 4. A separate authentication component can provide a single central authentication point for one or more portals and other Web resources. The authentication proxy can be implemented as a separate server machine, as a plug-in to the Web server or as a plug-in to IBM WebSphere Edge Server software.

In this setup, authentication is performed by a proxy server. The authentication proxy is the only node visible to clients connecting from the Internet. It is protected by an outer firewall. The authentication proxy cooperates with its authorization server, which is behind another firewall. This firewall allows inbound requests only from the authentication proxy and inbound responses from the outbound proxy. The outbound proxy forwards requests from the intranet and returns the responses from the Internet. The authorization server accesses the user registry, which is in the same network. The user registry is used by the authorization server, the portal server and by other systems.

The portal server functions execute on several machines in a cluster to provide a higher capacity and fault tolerance. In Figure 4, the cluster includes a load balance machine, several caching proxy machines and several portal server machines.

A portal database, content management server and search server are located in the same network as the authorization server, the user registry and the portal server cluster. An additional firewall separates the network from the intranet.

Another deployment possibility is based on a portal cluster, which uses a separate, load-balanced and fault-tolerant cluster of authentication proxy server machines (see Figure 5). The network node visible to the clients in the Internet is the load balancer, which dispatches requests between several authentication proxies. The authentication proxies are located in the DMZ, so that only authenticated requests pass through the second firewall into the production network.

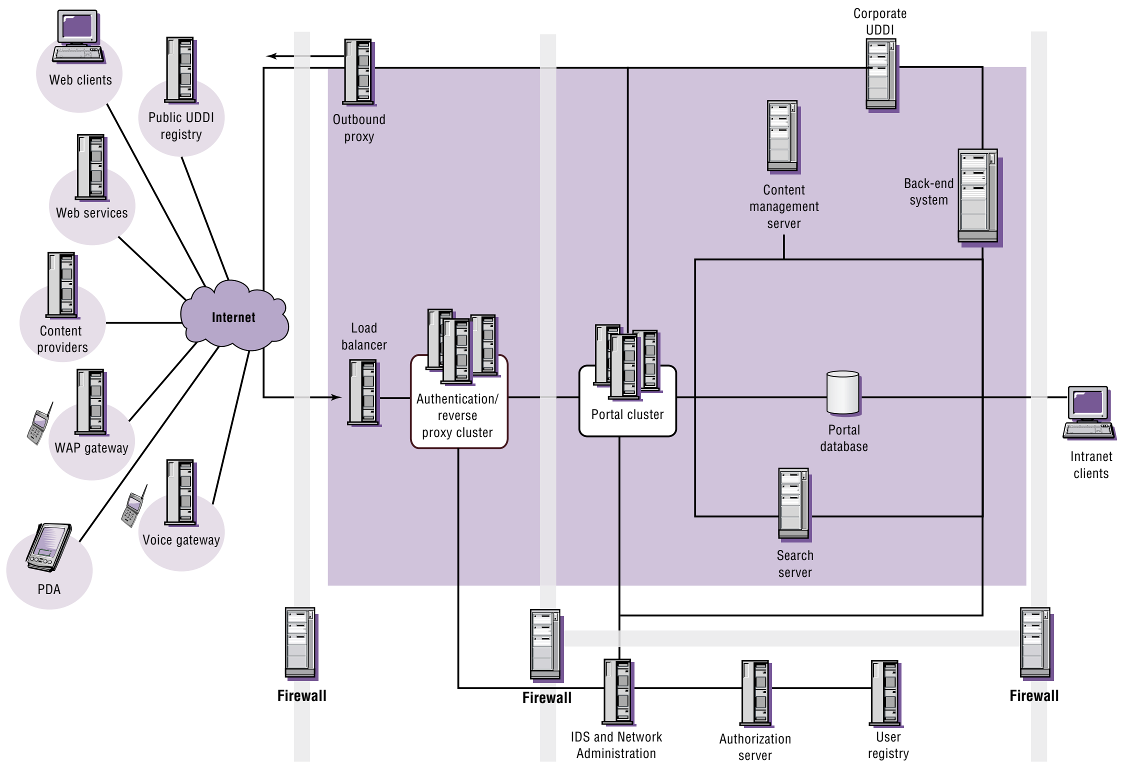

Figure 5. Portal deployment with clusters of portal servers and authentication proxies

The portal server cluster is placed in the production zone behind the second firewall with the portal database, the search server and the content management server. A third firewall separates the portal server cluster network from the intranet clients.

Figure 5 shows a typical network administration setup with a separate administration zone. The middle firewall uses a third network adapter with customized filters to securely connect the administration network with the outer DMZ and the production network. The authorization server, user registry, network administration system and Intrusion Detection System (IDS) are all within the adminstration network.

Available authentication methods

Access-control features require positive identification, or authentication, of the person or program requesting access to a secured property, such as to data or to a location. In most cases, the authentication process requires a user identification (user ID) string and a user password to verify the authority of the user. The standard way to provide the user ID and password to the server is through HTTP basic authentication[3], which uses a Web browser mechanism with a standard login dialog box. Alternatively, using HTTP form-based authentication, the server sends a customized authentication form to the user.

Password-based authentication schemes offer limited security, because passwords can be poorly chosen, shared, reused between secure and insecure systems or stolen. Other authentication mechanisms include SSL/TLS client authentication, based on digital signatures and certificates (including secure storage on smart cards), hardware-based authentication, and one-time passwords such as the RSA SecureID token. And you can use various biometric mechanisms, including fingerprint verification, iris scanning and voice recognition to improve authentication.

WebSphere Portal offers an authentication subsystem that delegates user authentication to underlying mechanisms of IBM WebSphere Application Server. The subsystem provides support for the following authentication setups:

- *Use of native WebSphere Application Server authentication. A custom login form posts the user's authentication data to a servlet that requests WebSphere Application Server security*[4] *to validate the user's authentication data. This setup exploits WebSphere Portal integration with WebSphere Application Server and its capability to configure the portal as a secure Web application.*
- *Use of an authentication proxy. WebSphere Application Server provides a trust association interceptor (TAI) interface that allows it to establish cooperation with trusted authentication proxies.*

Authentication through WebSphere Application Server security features

To use WebSphere Application Server security features, the portal is configured as a secure Web application. When WebSphere Application Server receives a request for the application (the portal), its security component requests the authentication credentials from the client. Depending on which authentication method is configured, the security component request will create an HTTP basic authentication request or an HTTP over SSL (HTTPS) client certificate request (which is an HTTP request used over a secure connection) to be sent to the Web browser. Or, the client is redirected to an authentication form that prompts the user to provide the credentials for authentication. In the latter case, the form posts the credentials to a WebSphere Portal custom authentication servlet that obtains the posted credentials. The servlet then makes the required calls to WebSphere Application Server security functions to log in the user to the WebSphere Application Server security environment. With either HTTP basic authentication or HTTPS client certificate authentication, WebSphere Application Server receives the authentication credentials directly. WebSphere Application Server authenticates users by checking provided credentials against a user registry. The user registry can be a Lightweight Directory Access Protocol (LDAP) directory or a custom user registry.

Authentication through a separate authentication proxy

An external authentication proxy can protect your portal by intercepting all requests targeted to portal destinations. An authentication proxy can be implemented as a proxy server such as WebSEAL in Tivoli Access Manager for e-business. Or it can be implemented as a plug-in to the HTTP server or to WebSphere Edge Server. Examples of HTTP plug-in authentication proxies include the Web Agent in Netegrity SiteMinder and the authentication plug-in in Entrust GetAccess. Tivoli Access Manager plug-in for WebSphere Edge Server plugs into WebSphere Edge Server instead of the Web server.[5] An external authentication component authenticates users by checking the provided credentials against a user registry, which is in most cases an LDAP directory.

Trust association interceptors registered with WebSphere Application Server establish a connection between WebSphere Application Server and the authentication component protecting it. TAIs are programs called by WebSphere Application Server to work with external authentication components, as shown in Figure 6. The program relies on external components to process authenticating requests rather than performing authentication directly. WebSphere Application Server defines the interface that a TAI uses to indicate that it can handle a request and that the request has been authenticated. TAIs communicate with the authentication component and make the authentication decisions accessible to WebSphere Application Server through the specified TAI interface.

After a request for a portal destination passes the external authentication component, it is received by WebSphere Application Server and sequentially passed to the registered TAIs until one TAI indicates that it is responsible for authentication and either accepts or rejects the request. When no TAI can handle the incoming request, WebSphere Application Server reverts to its native authentication, as if no TAIs are available. The client is then redirected to the custom login form. However, this only occurs with requests that bypass the external authentication component.

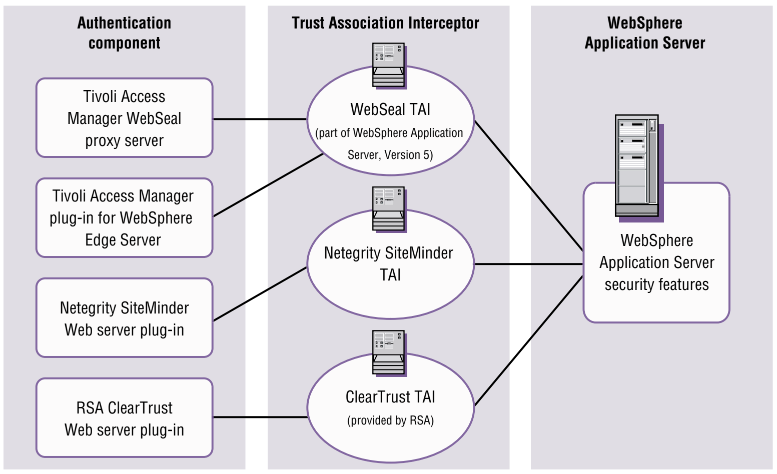
Authentication component
Trust Association Interceptor
WebSphere Application Server
Tivoli Access Manager WebSeal proxy server
Tivoli Access Manager plug-in for WebSphere Edge Server
Netegrity SiteMinder Web server plug-in
RSA ClearTrust Web server plug-in
WebSeal TAI (part of WebSphere Application Server, Version 5)
Netegrity SiteMinder TAI
ClearTrust TAI (provided by RSA)
WebSphere Application Server security features

Figure 6. Authentication components and TAIs

Figure 7 shows the detailed flow of control for a request that is passing through an external authentication proxy. The interactions are the same for authentication proxies implemented as separate servers and for proxies implemented as plug-ins for the Web server or WebSphere Edge Server.

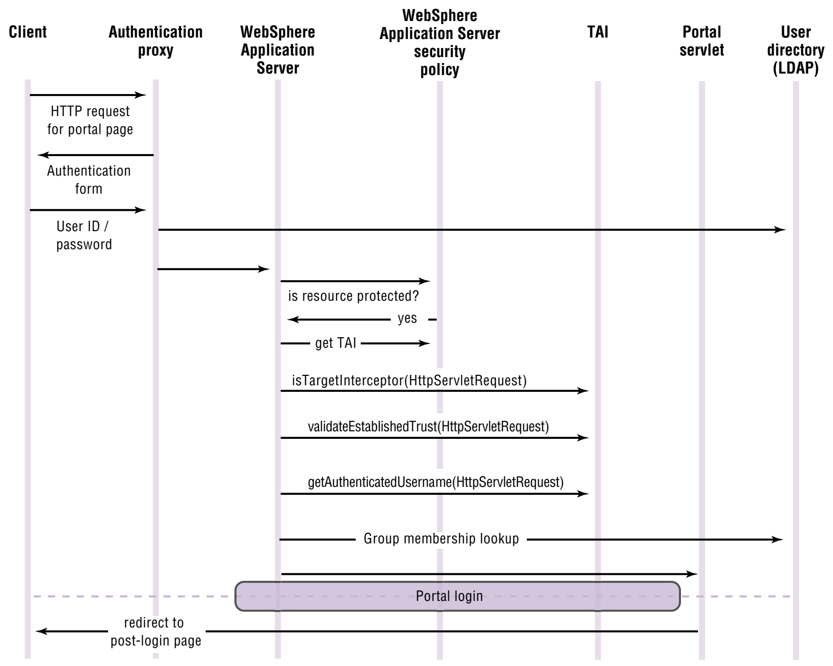
Client
Authentication proxy
WebSphere Application Server
WebSphere Application Server security policy
TAI
Portal servlet
User directory (LDAP)
HTTP request for portal page
Authentication form
User ID / password
is resource protected?
yes
get TAI
isTargetInterceptor(HttpServletRequest)
validateEstablishedTrust(HttpServletRequest)
getAuthenticatedUsername(HttpServletRequest)
Group membership lookup
Portal login
redirect to post-login page

Figure 7. Flow of control for a request passing through an external authentication proxy

Requests in an authenticated session

When authentication is complete, the user is logged in, a portal session is started and a Lightweight Third Party Authentication (LTPA) token, containing the user ID and an expiration date and time, is issued to the client, along with the HTTP session cookie (usually named JSESSIONID)[6]. LTPA is supported by WebSphere components.

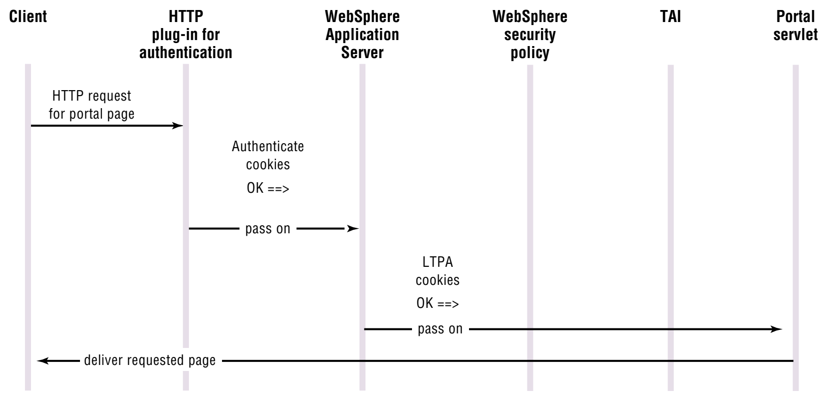
Client
HTTP plug-in for authentication
WebSphere Application Server
WebSphere security policy
TAI
Portal servlet
HTTP request for portal page
Authenticate cookies
OK ==>
pass on
LTPA cookies
OK ==>
pass on
deliver requested page

Figure 8. Flow for a request that is already authenticated

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 the LTPA single sign-on (SSO) domain (see Figure 8).

User registry versatility

A user registry is a repository that holds information about registered users and groups, as well as applications that are validated through the user registry. WebSphere Application Server and WebSphere Portal allow an internal WebSphere Portal database, an LDAP directory or a custom registry to be used as a user registry. WebSphere Portal shares the authentication registry with WebSphere Application Server while having a separate database for user profiles and preferences. Some profile information may also be stored in the same physical store as the authentication registry. For example, an LDAP directory may contain significantly more information about each user than simply a name and password.

In WebSphere Portal, Version 5.0, user information is centralized in the member manager component. This component in WebSphere Portal can be configured for different layouts of data in the user registry and its database as shown in Table 1. WebSphere Portal can also work with a read-only user registry so all portal user data to be updated is stored in the internal database.

Table 1. Supported authentication registries and corresponding WebSphere Application Server and WebSphere Portal settings

Member manager configuration	WebSphere Application Server authentication registry	Description
LDAP and database	LDAP	The authentication registry is a directory server. You can configure which profile information is stored in the LDAP and which is stored in the database.
Database only	Custom user registry provided by WebSphere Portal	WebSphere Portal provides a custom user registry implementation for the internal database. The authentication registry is part of the member manager component, and profile information is stored in the same database.
Customer-provided registry	Custom user registry provided by customer	The customer supplies a custom user registry. Profile information is held in the member manager component database as well as in the custom registry.

The member manager component determines group membership. This information is used by the WebSphere Portal access control and administration functions. The lookup function helps to evaluate nested groups. A configurable option stops group membership lookup at the first level to help prevent a decrease in performance.

Portal session login

After WebSphere Application Server authentication, the portal login is performed (see Figure 9). The user object is populated from the user registry and the user session is resumed from the saved state, if this option is selected. Finally, the Web browser is redirected to the target page, which can be defined as part of the redirection policy.

During the WebSphere Portal login sequence, a Java Authentication and Authorization Services (JAAS) login is executed. JAAS is a Java API that specifies classes in three different categories: common, authentication and authorization. WebSphere Portal primarily uses the common classes, such as the JAAS subject. The authentication classes—such as LoginContext and LoginModule—are used only with partial functionality, and the authorization classes remain unused. A JAAS login, as defined by the JAAS specification, executes a number of login modules and returns the user's subject as a result of a successful login. The subject is a container for the user's identities (principals) and credentials. Each login module can add principals or credentials to the subject.

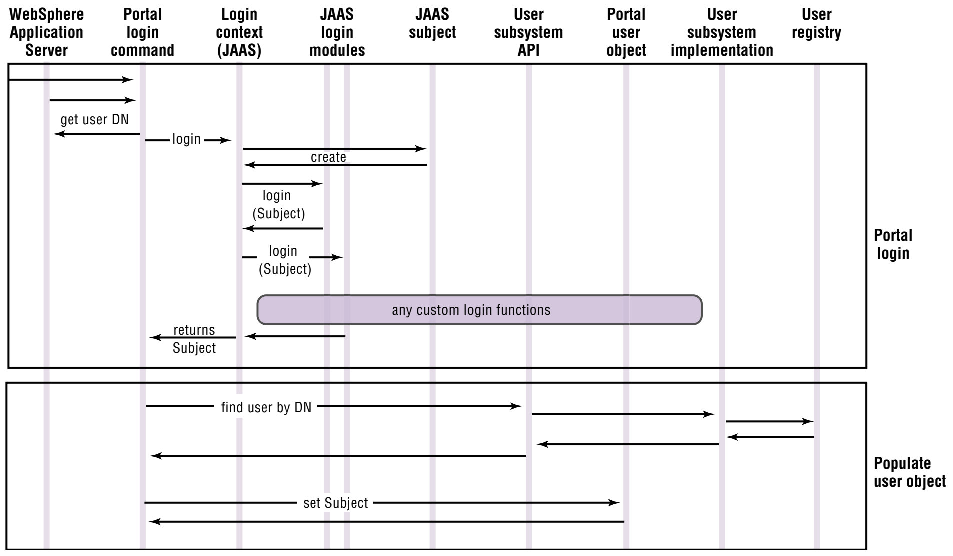

Figure 9. Portal login after successful authentication

By specification, JAAS login modules may also try to authenticate the user. In WebSphere Portal, however, user authentication is already performed by the authentication component. So, WebSphere Portal JAAS login modules typically only populate the user's JAAS subject, which allows you to store, for example, SSO tokens. These tokens can be used by special active credential objects that enable portlets to access other applications within the SSO domain.

User logout or timeout

A session can timeout at a specified interval of inactivity, or users can log out manually with a logout button added by the aggregation engine to every page banner. In the event of inactivity, when a timeout occurs, the portal logout function performs the following actions:

- *Suspend User Session. The user's portal session is persisted.*
- *Portlet user logout. The portlets are notified of the event "user logged out" to give them the option to finalize or terminate active actions and transactions.*

If the logout was initiated intentionally by the user and not by session timeout, a WebSphere Application Server logout is performed. The user's credential token is marked as invalid, and a respective cookie invalidation command is added to the response. The browser is then redirected to render the post-logout target. When using an authentication proxy, WebSphere Portal must be set up to redirect to the authentication proxy's logout page after logout. That way, the authentication proxy is also notified of the logout.

Customization of the portal login and logout process

WebSphere Portal allows you to customize the portal login and logout procedures. The following modifications are possible:

- *Specifying the post-login and the post-logout redirection policy and targets*
- *Adding custom JAAS login modules that are used to authenticate a user and populate the portal's user JAAS subject with principals or credentials*
- *Securing the login interactions and the personalized portal pages through SSL*
- *Extending or replacing the portal login and logout action classes*

Single sign-on capabilities

You can use WebSphere Portal to integrate enterprise information systems and present them through the portal user interface. You want back-end systems to perform authentication and authorization separately – to maintain current security controls – without repeatedly prompting the user to authenticate. The single-sign-on capability provides a reliable authentication method for one user at a time. It allows, within a single environment – and using a single authentication for the duration of the session – access to other applications, systems and networks.

With WebSphere Portal, there are two SSO realms, one from the client to portal and other Web applications. The other from the portal to back-end applications (See Figure 10). With single sign-on from the client to Web applications and the portal, a client logs in once to one Web application and is then able to access all other Web applications that are part of the same SSO realm without a second authentication challenge. It doesn't matter whether WebSphere Portal is the authenticating Web application. Similarly, single sign-on from the portal to back-end applications allows a portal client to log in to the portal once and then access a number of back-end applications through respective portlets without having to authenticate at each of these applications.

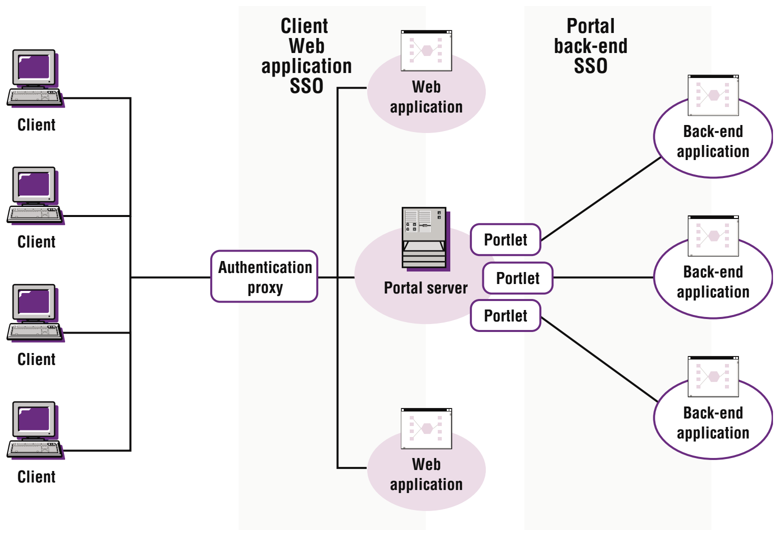

Figure 10. WebSphere Portal SSO realms

Single sign-on from a client to Web applications

Single sign-on from a client to Web applications and a portal can be provided through a variety of mechanisms:

- *LTPA support built into WebSphere Application Server*
- *Authentication proxy single sign-on support (such as WebSEAL, Netegrity SiteMinder, RSA ClearTrust and Entrust GetAccess)*

In function, the various SSO solutions are essentially the same. With successful authentication, the client receives a security token that enables it to prove successful authentication and to pass subsequent requests through without repeating the authentication.

Persistent cookies can be used to achieve single sign-on. The Web browser stores persistent cookies on the client in a form that has little protection against reading and copying. For greater security, however, WebSphere Portal only uses cookies that are deleted when the Web browser session is terminated – not persistent cookies.

Single sign-on between Microsoft Windows workstations and portals

With IBM Tivoli Access Manager for e-business, Version 4.1 and later, you can realize single sign-on between a Microsoft® Windows® workstation and portals. This means that a user can log on to a local workstation and then use Microsoft Internet Explorer to access a portal without having to authenticate identity separately at the portal page.[7]

Single sign-on from portal to back-end system

WebSphere Portal, Version 5.0 offers a credential vault as a portlet service. This service provides the portal and portlets with a mechanism to map from one user identity—usually a user ID (principal)—to another user identity with a credential, such as a password. With this capability, you should not use portlets to store user credentials as part of the user-specific portlet data. In fact, storing credentials in the portlet data is no longer recommended to achieve a single sign-on. You should update portlets that store credentials to make use of the credential vault (see Figure 11).

The credential vault service provides the following functions:

- *Map the requested credential slot, the user ID and the portlet ID to a resource in the vault. A portlet can only retrieve a credential if a respective mapping rule exists. Each credential slot is associated with a certain vault implementation (the actual store), which allows different credentials to be kept in different physical stores.*
- *Retrieve the user's credential. Some credentials will be stored and managed by the portal that always uses the local default vault store. If a user password is not stored in the portal's local vault, it will be acquired from the respective external vault.*
- *If a credential is not available or the authentication fails, an appropriate exception is thrown. The service passes this exception to the portlet to allow appropriate error handling. For example, a prompt may require the user to set the credential through the portlets edit mode.*
- *The credential vault will not allow anyone but the credential owner—not even the portal administrator—to manage or use the credentials, which preserves the trust of the end user. No method to access another user's credentials will be provided.*
- *Administrate the credential vault (vault-management interface). Portal administrators can configure the credential vault services that are not controlled by the user. This includes the management of the vault segments, the administration defined slots and system (shared) credentials. Although an administrator cannot access or change user-defined credential slots and passwords, user-defined slots are deleted if the respective user is deleted.*

Figure 11 illustrates the structure of the WebSphere Portal credential vault. The credential vault is organized as follows:

- *The portal administrator can partition the vault into several vault segments. Vault segments can only be created and configured by portal administrators.*
- *A vault segment contains one or more credential slots. Credential slots are the "drawers" from which portlets retrieve credentials and to which they are stored. Each slot holds one credential, if user-defined, and one credential per user, if defined by the administrator.*[8]
- *A credential slot is linked to a resource in a vault implementation, where the credentials are actually stored. Examples for vault implementations are the WebSphere Portal default database vault or the Tivoli Access Manager repository.*

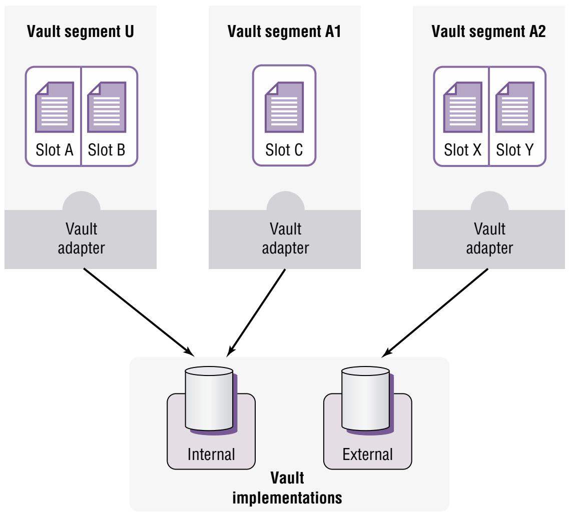

Figure 11. WebSphere Portal, Version 5.0 includes a credential vault portlet service

For each vault segment, a flag indicates whether it is to be managed by the administrator or by the user. Only administrators can create credential slots in an administrator-managed vault segment. Portlets (acting on behalf of a portal user) are permitted to create credential slots only in user-managed vault segments, but they can set and retrieve credentials in both types of segments.

Therefore, portlets that need a credential to complete their service have two options: to use an existing credential slot that has been defined by the portal administrator in an administrator-managed vault segment or to create a new credential slot in a user-managed vault segment.

The WebSphere Portal credential vault service distinguishes between three types of credential slots:

- *A system credential slot stores system credentials. Credentials are shared among all users and portlets.*
- *A shared credential slot stores user credentials that are shared among the user's portlets. Credentials are user-specific but the same for all portlets of a user.*
- *A portlet private slot stores user credentials that are not shared among portlets. Credentials are user-specific and specific to an individual portlet instance.*

The credential vault service returns credentials in the form of credential objects. WebSphere Portal differentiates between passive and active credential objects.

Passive credential objects are a container for a credential's sensitive information, such as the user ID and password. Portlets that use passive credentials must extract the information out of the credential. With passive credentials the portlets are responsible for the communication (authentication) with the back-end system, shown in Example 1.

Passive credential object use (pseudo code)

```
// Retrieve the credential object from the credential vault
UserPasswordPassiveCredential credential = (UserPasswordPassiveCrede
ntial)service.getCredential(slotId, "UserPasswordPassive", null,
porletRequest);

// Extract the actual secret out of the credential object

UserPasswordCredentialSecret secret = credential.getUserSecret([…]);

// Portlet connects to back-end system and authenticates using the
user's secret....

// Portlet uses the connection to communicate with the backend appli-
cation....

// Portlet takes care of logging  at the back-end....
```

Example 1. How a portlet uses a passive credential that carries a user ID and password

An active credential object hides the user ID and password from the portlet, making it an ideal method to preserve portal security. Active credentials allow portlets to trigger authentication to remote servers using standard mechanisms – such as HTTP basic authentication, HTTP form-based authentication or POP3 authentication. To access server-side credentials, active credentials allow the use of hardware tokens that never expose their sensitive information. Such hardware tokens[9] only allow the use of credentials inside the hardware with the supported authentication algorithms.

Example 2 illustrates how a portlet uses an active credential. In this case, it's an active credential for HTTP form-based authentication, shown in Figure 12.

Active credential object use (pseudo code)

```
// 1. Retrieve the credential object from the credential vault
HttpFormBasedAuthCredential credential = (HttpFormBasedAuthCredential)
service.getCredential(slotId, "HttpFormBasedAuth", config, request);

// 2. Log into the backend system credential.login();

// 3. Get an authenticated connection to use

URLConnection connection = credential.getAuthenticatedConnection();

// 4. Portlet uses the connection to communicate with the backend
application...

// 5. Log out of backend system credential.logout();
```

Example 2. How a portlet uses an active credential

All credential types that are available within the portal are registered in a credential type registry. WebSphere Portal, Version 5.0 provides a small set of credential types out of the box, but additional credential objects can easily be registered.

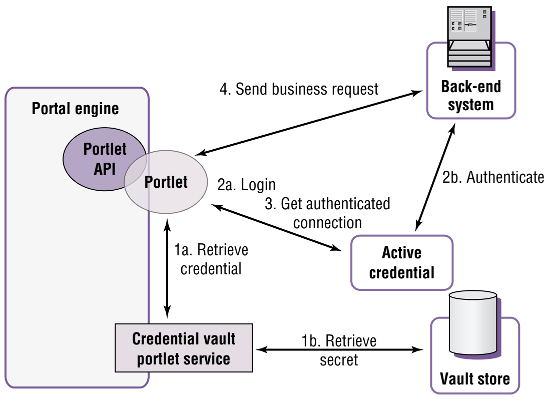

Figure 12. A portlet using an active credential object for back-end single sign-on

Included passive credential objects:

- *SimplePassive. Stores credentials as "serializable" Java objects.*
- *UserPasswordPassive. Stores credentials as user ID-password pairs.*
- *JaasSubjectPassive. Stores credentials as javax.security.auth.Subject objects. It is used to provide portlets with the JAAS subject that the portal established for the user.*
- *BinaryPassive. Stores credentials as a byte array (simple implementation of PassiveCredential for binary credentials).*

Included active credential objects:

- *HttpBasicAuth. Stores user ID and password and provides support for HTTP basic authentication.*
- *HttpFormBasedAuth. Stores user ID and password and provides support for HTTP form-based authentication.*
- *JavaMail. Stores user ID-password pairs and leverages the authentication functionality of the javax.mail API.*
- *LTPA token. Supports authentication at a back-end system that is within the same WebSphere Application Server SSO domain as the portal.*
- *SiteMinderToken. Supports access of back-end systems that are in the same SiteMinder SSO domain as the portal. It is typically used when a SiteMinder authentication proxy protects the portal and other resources.*
- *WebSEALToken. Supports access of back-end systems that are in the same WebSEAL SSO domain as the portal. It is typically used when a WebSEAL authentication proxy protects the portal and other resources.*

For security reasons, credential objects do not implement java.io.Serializable and can therefore only be stored in the portlet session as a transient value. The credential classes store the actual credential secret as private attributes. If they were serialized into the WebSphere Application Server session table, the credential could, potentially, be read by anyone who has access to this database.

Client-to-portal communication security
SSL and TLS protocols[10] leverage different cryptographic algorithms to implement security—for example, authentication with certificates, session key exchange algorithms, encryption and integrity check. They are commonly used to help provide privacy and reliability between communicating applications such as Web clients and Web servers. SSL and TLS provide connection security with three basic properties:

- *Confidentiality. Encryption is used after an initial handshake to define a private key. Symmetric cryptography such as Data Encryption Standard (DES) or RC4 is used for data encryption.*
- *Authentication. User identity can be authenticated using public key cryptography such as RSA or DSS.*
- *Integrity. Message transport includes a message integrity check using a keyed message authentication code (MAC). Secure hash functions such as SHA and MD5 are used for MAC computations.*

IP security standard (IPSec) technology 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 are preconfigured, so it should not be used for 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 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 could exploit crypto hardware as well, but each application would have to support this. SSL, on the other hand, is widely supported and administrators gained a lot of experience and confidence with this technology over the years.[12]

Whether using SSL or network layer security, you need to decide where to terminate the protected connection and what communication to protect. It's necessary not only to protect the communication across unprotected networks (such as the Internet), but also communication within corporate networks. You should not underestimate the possibility of human error or internal attack.

Figure 13 shows how SSL can be implemented in the simplest deployment scenario (see Figure 2). With no systems or firewalls between the client and portal server, one connection has to be secured with SSL.

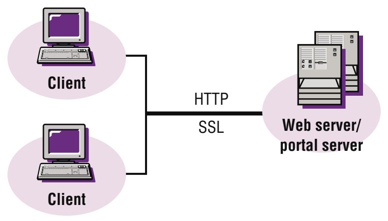

Figure 13. Communication security for portal installations within a protected network

In the second deployment scenario (see Figure 3), the Web server is separated from the portal server with firewalls between client, Web server and portal server. For this example (see Figure 14), the DMZ is not fully trusted, but communication is allowed to be unprotected within they production network. Again, SSL must be used for the client connection. The first firewall is configured to allow this kind of data to pass through. The path between the Web server and second firewall is secured using IPSec.

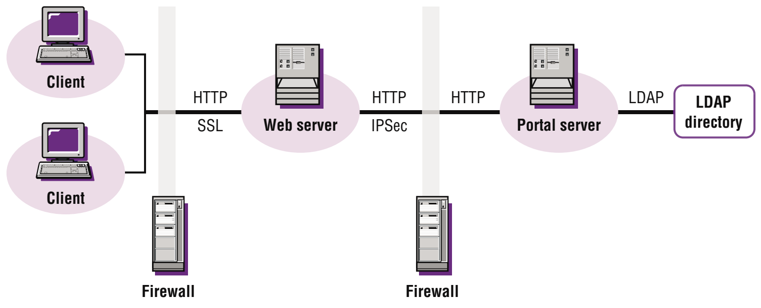

Figure 14. Communication security for portal installation in a trusted intranet

The third deployment scenario (see Figure 4) places a reverse proxy for authentication in the DMZ and adds a cluster of portal servers – fed by a load balancer – behind another firewall. A possible secure communication implementation could use SSL from the client to reverse proxy, and IPSec for the remaining path between the servers (See Figure 15).

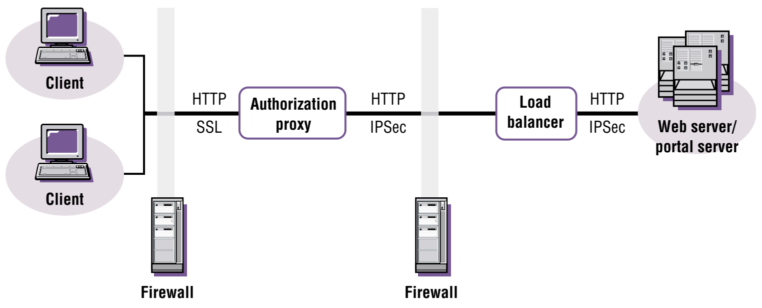

Figure 15. Communication security for an Internet portal cluster

In a fully clustered portal setup with all communication protected (see Figure 16), the client's SSL connection is tunneled through the first load balancer and terminated at the authentication proxy cluster. This tunneling is allowed because the first load balancer doesn't have to read the request for its load-balancing strategy. The second load balancer, however, will typically use such a strategy and therefore the next SSL connection needs to be terminated at this server. Lastly, a third SSL connection exists between the load balancer and portal cluster. When an LDAP directory is added to the deployment scenario (see Figure 16), communication between the application server and LDAP server can be protected because the LDAP queries and responses might contain confidential data. The WebSphere Application Server security component can be configured to use LDAP over SSL (LDAPS).

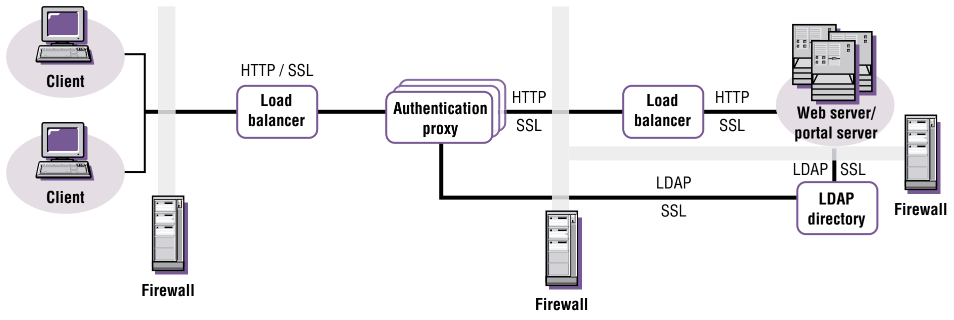

Figure 16. Communication security implementation in a fully clustered portal setup

Restricting protection to sensitive communication only

Using SSL consumes 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, it is recommended that you keep SSL protection at the necessary minimum. WebSphere Portal allows you to switch between SSL and non-SSL connections during a session. It's possible to set up a portal so that SSL is used for user login only, for login and all personalized content or for the entire portal. The final decision whether a portal should protect none, all or only parts of its communication with SSL should be based on the sensitivity of the data.

Using client certificates

The SSL and the TLS protocol specify the authentication of communication partners through the use of X.509 certificates. The client may use a certificate to authenticate to the server. This type of authentication through the client certificate is more convenient than typing a password. And it avoids the known security drawbacks of passwords. In addition, if the private key associated with the client certificate is stored on a smart card, this form of authentication can establish particularly strong trust.

Enforced certificate-based client authentication is a configuration option with WebSphere Application Server that must be selected during configuration if an authentication proxy such as Tivoli Access Manager WebSEAL is used. WebSEAL offers additional control, called step-up authentication. You can configure it to require and automatically prompt for different forms and strengths of authentication for different targets.

Secure back-end connections

Some portlets are designed to access back-end applications that house critical enterprise data. WebSphere Portal can help keep your data private in two ways:

1. *Selected portlets can use SSL connections to exchange data with a corresponding back-end application. Depending on the back-end application, an SSL handshake with client authentication may be required.*
2. *The portal and the back-end applications can establish a virtual private network (VPN) using IPSec.*

WebSphere Portal offers a content access portlet service that enables portlets to establish and use an SSL connection. It is designed to facilitate assigning the portal a collection of client certificates that portlets can use to self-authenticate to back-end applications. This collection may be different from what the application server uses to authenticate directly. The portal certificates are only available to the portal and its portlets. Other Web applications that are installed on the same application server node cannot use them.

The content access service offers extensive SSL functionality:

- *If an HTTPS protocol handler has been configured, any component (particularly portlets) can initiate an HTTPS connection with URLConnection, HttpURLConnection or HttpsURLConnection. The SSLContext and SSLSocketFaktory will be created and assigned to the connection object on the fly. The portlets can either request a markup or an input stream, and no other communication with the content access service is necessary.*
- *If an HTTPS proxy has been configured, any HTTPS URL requested from content access service will go through that proxy, unless it is explicitly listed on the content access service exclusion list. Ports other than 443 can be configured.*
- *An SSL key store, a collection of keys that can be used to verify the portal's identity to remote systems, can be configured. The key store will be used for all direct HTTPS requests initiated by content access service but not for the SSL handshake between an HTTPS proxy and the back-end application.*

- *An SSL trust store, a collection of trusted certificates that remote systems may present to the portal to self-authenticate, can be configured. The trust store will be used for all direct HTTPS requests initiated by the content access service but not for the SSL handshake between an HTTPS proxy and the back-end application.*
- *If an HTTPS URL returns an HTTP redirect and the maximum number of redirects has not been exceeded, the content access service will follow the redirect using the HTTPS protocol and the HTTPS proxy, if indicated.*

Using IPSec for back-end connections is an alternative to using SSL. IPSec operates efficiently on the network layer and is easy to configure in most current operating systems.

Understanding access-control concepts to protect portal resources

Authorization is dependent on authentication, because verifying that the user is trusted determines whether you allow access. WebSphere Portal provides a facility to manage access to portals—whether to an entire portal or to specific parts, such as individual pages. You can use portal access-control capabilities to define the operations each user is allowed to execute within the portal object space. This set of operations represents the total permissions granted to the user. WebSphere Portal uses roles, based on the job responsibilities of the individual users, to enable convenient permission management. Assigning a role grants all of its permissions to the user. You can administer access control using corresponding portlets within the running portal or through the XmlAccess scripting interface.

Sensitive operations

Portal users interact with portals in various ways, potentially triggering the execution of a high number of different operations within the portal object model. Access to almost all of those operations has to be restricted to a well-defined set of users. Users are only allowed to execute the sensitive operations for which they have sufficient permissions.

The WebSphere Portal access-control policy maps each sensitive operation to a specific set of permissions that are required to gain access to the operation. Access-control administrators then define which users have which permissions by creating or deleting the corresponding roles and role assignments. Sensitive operations defined within WebSphere Portal include simple operations, like "view a specific portlet on a specific page", and more complex tasks, such as "install new Web module" or "run XmlAccess scripts."

Roles and role types

To allow a user or a group of users to execute specific, sensitive operations, you have to assign the roles that provide the necessary permissions. Role assignments can be either explicit (if a user or user group is directly assigned to a role) or implicit (if the user or user group is a member of a user group that has a corresponding explicit role assignment). The set of permissions granted to a specific user is defined as the total of all permissions contained in all explicitly and implicitly assigned roles of the user.

Each role within portal access control is an instance of a specific role type. WebSphere Portal, Version 5.0 supports a range of role types, including Administrator, Security Administrator, Delegator, Manager, Editor, Privileged User and User. Role types model the different ways people interact with portals, depending on their job responsibility:

- *Users can view portal content, such as a portal page.*
- *Privileged Users can view portal content, personalize portlets and portal pages and create new, private pages.*
- *Editors can create new, shared resources and configure existing resources used by multiple portal users.*
- *Managers can create new, shared resources, and configure and delete existing resources used by multiple portal users.*
- *Administrators have unrestricted access to create, configure and delete shared resources. And, Administrators can grant access to resources by creating or deleting corresponding role assignments.*
- *Security Administrators can create and delete role assignments for roles paired with specific resources.*
- *Delegators can assign specific users or user groups to roles.*

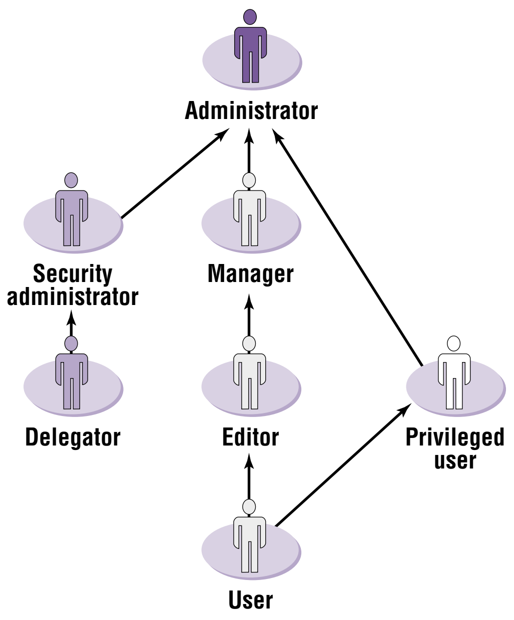

Figure 17. In the role type hierarchy, role types are organized by hierarchy. Each greater role type offers more access while including the privileges of lesser role types.

Role assignments and inheritance

A specific role is not only characterized by its role type, but also by its domain root resource. Roles are named with the convention <role type>@<domain root resource>. For example, the role Editor@Market_News_Page names a role of type Editor tied to a specific page in the portal page hierarchy (see Figure 17). Permissions allow users with the Editor role – or with any role type below the Editor role type in the role type hierarchy to execute sensitive operations on the Market_News_Page.

Roles not only contain permissions that allow users to access the domain root resource. Roles also inherit corresponding permissions on all descendant resources under the domain root in the resource hierarchy. You can prevent permission inheritance by creating role blocks.

Role blocks

Sometimes you don't want a specific role to affect all the resources below it in the resource hierarchy. To prevent this, you can switch off inheritance at a specific resource. For example, as illustrated in Figure 18, a role block assigned to Europe_Market_News_Page prevents the Europe_Market_News_Page from inheriting access-control configuration from its parent page.

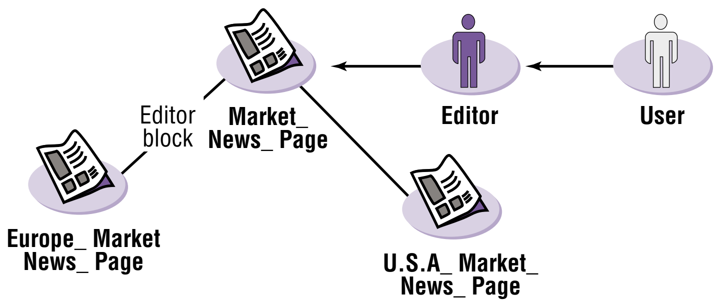

Figure 18. Permission inheritance allows users with a role assignment to the Editor@Market_News_ Page role to act in the Editor role on the USA_Market_Page.

Role blocks are role-type specific and tied to a specific resource. This means that a specific role block prevents inheritance on the corresponding resource for all roles of the corresponding role type. As shown in Figure 18, a block for the Editor role type at the Europe_Marketing_News_Page does not block other role types, such as Manager or User. You can block each role type separately as necessary. Administrator or Security Administrator roles can't be blocked.

WebSphere Portal supports two kinds of role blocks. Inheritance blocks (above the resource) shield a resource from inheriting from parent resources, as in Figure 18. And propagation blocks (below the resource) prevent inheritance from parent to child resources.

Ownership

With WebSphere Portal, you can give access to a resource by identifying a dedicated owner. Usually, when a user creates a new portal resource, such as a new page, the executing user automatically becomes the initial owner of that resource. WebSphere Portal gives the owner of a resource specific permissions, including the right to delete the resource. Ownership provides the same permissions as the Manager role. Resource ownership cannot be inherited, but it can be changed. You can assign a specific resource a new owner, effectively giving the new user the corresponding privileges of the original owner.

Private pages
WebSphere Portal — through portal access-control functions — supports the creation of private pages, which can be accessed only by the owner of the page. Privileged Users can personalize a nonprivate page and save their changes as a new, private page. And because the new page is owned by the Privileged User, only that user can access the private page — a copy of the original, modified to their preferences.

Protected resource hierarchy
WebSphere Portal uses virtual resources two ways. First, virtual resources guard sensitive operations that affect the entire portal, instead of specific resource instances. For example, WebSphere Portal uses the virtual resource XmlAccess to protect the ability to execute XmlAccess scripts on a portal. Second, WebSphere Portal uses virtual resources to group resources of the same or related resource types. The virtual resource *pages* is the root node of all pages (resource instances of the *page* resource type) within the portal page hierarchy.

Resource instances are structured according to the portal object model. This means that portlets are represented by child resources of their containing portlet applications, and subpages are modeled as child resources of their corresponding parent pages. The WebSphere Portal access-control function protects Web modules, portlet application definitions, portlet definitions, content nodes (pages, labels and external URLs), URL mapping contexts, user groups and users (only through the groups to which users belong).

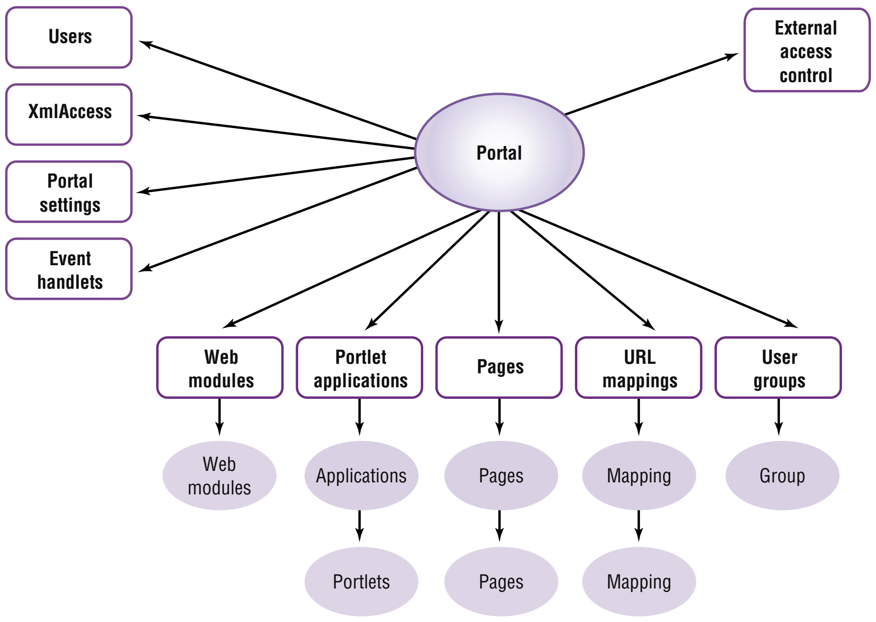

Figure 19. All portal resources—whether actual or virtual—that are potentially affected by sensitive operations are part of a protected resource hierarchy.

Administering access control

With WebSphere Portal, you can administer access control through corresponding portlets provided within portal administration facilities and using XmlAccess. There are two portlets that provide two different views on the actual portal access-control configuration. The resource permissions portlet allows an administrator to navigate through the protected object space and provides a resource-oriented view of the configuration and corresponding update facilities. The user-and-groups permissions portlet allows an administrator to select a specific user or user groups and provides a corresponding view—whether for users or user groups—in the portal access-control configuration. A complete set of configuration facilities is supported by XmlAccess—providing full scripting capabilities to configure portal access control.

Delegating administration

WebSphere Portal access-control functions enable you to delegate administration privileges. Access-control administrators are users who can change access control configuration, such as when creating role assignments. Delegated administration allows an administrator to delegate specific subsets of administrative privileges to other users or user groups. The receivers of these privileges become subadministrators, and can, in turn, delegate subsets of their privileges to others.

WebSphere Portal provides a delegated administration policy that controls which permissions are necessary to allow users to make changes to the access control configuration. This policy allows you to restrict administrative privileges to specific parts of the protected resource space and to specific user groups. You can establish sub-administrators to control role assignments for a specific branch of the portal page hierarchy and for a specific set of user groups.

When WebSphere Portal is installed, the original user is initially assigned the Administrator role at the root resource of the protected resource hierarchy (Administrator@Portal). This role contains a specific, unique permission which is only contained in the roles Administrator@Portal and Security_Administrator@Portal that allows unrestricted administration of portal access-control functions.

Integrating external authorization systems

You can configure WebSphere Portal to allow external authorization systems to control access to portal resources—while protecting your IT assets. WebSphere Portal, Version 5.0 includes authorization adapters for Tivoli Access Manager and Netegrity SiteMinder.

You can allow an external system to protect individual subtrees of the protected resource hierarchy. You simply select a corresponding root resource from the protected resource hierarchy, and set which resources will be protected internally or by the external authorization system. Inheritance always stops between resources that have different externalization states — even when accessed by administrators and security administrators. This means that each resource is either exclusively protected by WebSphere Portal access-control functions or by the external system.

Using external systems to protect resources doesn't mean that the resource information is added to the protected object space of the external system.
It simply means that only the roles that exist on those resources are registered there. An administrator of the external system can then create and delete role mappings for these roles. Using WebSphere Portal, you retain exclusive control of roles and role blocks. The external authorization system can only control the mapping between roles and users or groups.

WebSphere Portal limits the ability to change the status of resources between internal and external control by designating the change a sensitive operation. This requires role assignments to the roles Security_Administrator@Portal and Security_Administrator@External_Access_Control. External access control is a virtual resource that guards the concept of externalizing and internalizing. This resource is externalized automatically during WebSphere Portal installation and cannot be internalized. This way, an external security administrator can't be overruled by a WebSphere Portal security administrator.

Maintaining portal security

Your portlet developers are charged with keeping sensitive data safe. And they can't afford to allow potentially vulnerable areas of your portal system to be left unprotected. Even with built-in portal server security features, there are risks. The portal engine can't, by itself, protect against malicious portlets. Your administrators are responsible for installing only trusted portlets.

Some portlets require secure connections. You need to guarantee the data that portlets send is kept confidential. If the portal is enabled to serve pages through SSL, portlets can initiate an SSL connection (with a start transaction link). However, for each request, the portlet has to check whether the connection is still secure. This verification can be done with the request.isSecure() method. If a request does not come over a secure connection, the portlet must not write confidential data to the output stream.

Some portlets accept data from any user and pass those data on to a different user. If the portlet does not take special precautions to guard against malicious input, the receiving user's application (their Web browser) might process that data with unfavorable results. This is known as a "cross-site scripting attack" because input malicious source is usually executed as scripting language by the receiving user. In order to prevent such attacks, the portal engine filters all input and converts the less-than (<) and greater-than (>) characters to the respective HTML escape sequences.

A similar problem can arise when a portlet aggregates markup language code from a third-party server – for example, with a clipping portlet. Unfortunately, the portal engine isn't able to filter markup commands accurately and can't offer protection. This means your portlet developers have to make sure that potentially malicious markup is filtered out of the aggregated data.

Summary

WebSphere Portal offers state-of-the-art protection through industry-standard security protocols and cryptographic algorithms. Single sign-on capability at the portal frontend and for back-end applications provides a superior user experience. The portal server can be flexibly integrated with existing corporate user directories and with products for authentication, authorization and administration. In addition to password-based authentication, stronger methods like X.509 certificates or one-time passwords (with WebSEAL) are supported. Secure communication can be used wherever the transmitted information is sensitive.

For more information

To learn more about IBM WebSphere Portal, Version 5.0, visit **ibm.com**/websphere/portal.

1 IBM WebSphere Application Server, Version 5 Security, **ibm.com**/software/webservers/appserv/was/library/

2 Understanding the Portlet Component Model in IBM WebSphere Portal, visit http://www7b.boulder.ibm.com/vadd-bin/ftpdl?1/vadc/wsdd/pdf/WebSpherePortalandPortlets.pdf

3 HTTP basic authentication has a known disadvantage: The browser caches a user ID and password and sends both with every request to the same target. Because there is no standardized mechanism through which the browser will be notified of the user's logout, the browser will continue to implicitly log in the user as soon as the same target is accessed again.

4 IBM WebSphere V5.0 Security WebSphere Handbook Series, SG24-6573-00, visit http://www.redbooks.ibm.com/redbooks/pdfs/sg246573.pdf

5 Plugging into WebSphere Edge Server offers scalability advantages.

6 For devices that cannot manage cookies (like WAP phones), proxy gateways are used to handle cookies.

7 Requires Microsoft Internet Information Server (IIS) to host the portal and use of Tivoli Access Manager Web server plug-in for IIS.

8 WebSphere Portal, Version 5.0 installs with one preconfigured, user-managed vault segment and does not allow defining any additional user-managed segments. Administrator-managed vault segments can be created without any limitation.

9 IBM has a cryptographic coprocessor (IBM 4758PCI) to store the private key and execute the cryptographic functions. This module is designed to meet the FIPS PUB 140-1 level 4 specification. The card keeps the server's private keys securely (any data tampering will be recognizable) and executes encryption algorithms. The private keys do not leave the card.

10 While TLS is the standard implementation, SSL is still more widely distributed. In this document, when SSL is used, TLS is implied.

11 Read "SSL/TLS in WebSphere: Usage, Configuration, and Performance," available at **ibm.com/**developerworks/websphere/zones/portal/

business software

G325-2090-02