Best practices for access control in multi-tenant cloud solutions using Tivoli Access Manager

Provide tenant awareness and single sign on while protecting your application resources

Christina Lau
Thomas Mikalsen
Bhadri Madapusi

May 01, 2011

In this fourth article on best practices for building multi-tenant applications in the IBM SmartCloud Enterprise, learn how to use the IBM Tivoli Access Manager to provide tenant awareness, protect your application resources, and to provide single sign-on.

Today cloud platforms and cloud-based solutions are a reality. More and more businesses are taking advantage of hosted software, solutions and services, outsourcing non-core competencies and cutting costs. In the course of building out several Software as a Service (SaaS) applications, the authors used the IBM Tivoli Access Manager as a shared component in the IBM Cloud to satisfy many requirements around security. This article describes the crucial role that Tivoli Access Manager plays in enforcing security between tenants.

Introduction

Multi-tenancy is an approach to support multiple customers within a single, shared application environment. To further exploit economy of scale, it is also desirable to support multiple SaaS applications within a single, shared SaaS delivery platform. The SaaS delivery platform itself becomes multi-tenant enabled, where its tenants are multi-tenant applications. Within such a platform, enforcing security constraints between applications and between application tenants is a critical requirement and Tivoli Access Manager can play a key role. Figure 1 shows the overall Tivoli Access Manager components running in the IBM Cloud and their relationships with other software components.
Figure 1: Tivoli Access Manager running in production environment on the IBM Cloud

Specifically:

- Tivoli Access Manager WebSEAL Server: A security proxy component that enforces access to web servers and manages the web space centrally, linking all web servers into one logical web space.
- Apache Server: Acts as a reverse proxy and distributes the load from incoming requests to the SaaS applications. Rewrite URL in each incoming request to match the relevant internal location of the requested resource.
- Tivoli Access Manager Policy Server: Supports policies definition and security administration based on the policies. For example, password rules.
- Tivoli Access Manager LDAP (Lightweight Directory Access Protocol): A directory server that stores information about its authorized users and what privileges each user has.
- Tenant management server: Used to manage information about a SaaS offering, the companies and users that are subscribed to the offering, the number of editions or seats within an offering, their pricing etc.
- SaaS application servers: Cluster of application servers that are specific to each SaaS application.
- The vertical lines represent firewalls configurations. Firewalls in this picture emulate the traditional demilitarized zone (DMZ) setup for corporate firewalls between the intranet and Internet. They are intended to add additional IP level access control.

Tivoli Access Manager for e-business is a mature product that provides robust, policy-based security to a corporate web environment. This includes authentication of users, control of access privileges, auditing, single sign-on, high availability, and logging that are elements of any security management solution. There are many excellent documents that describe its capabilities. In the following scenario, we focus on areas where it is specific to the cloud and tenant management.

Coarse-grained access control

In general, you can classify application resources (for example a URL such as http://www.mycompany.com/logo.gif) as either unprotected or protected. Unprotected resources are
accessible by any user and do not require authentication. Examples of unprotected resources can be static images, login pages, registration pages, demos, marketing collateral etc.

Protected resources are only accessible by a user after the user is authenticated and the details about the user such as their roles, what organizations or teams they belong to are established. Examples of protected resources can be user profiles, documents, emails, application widgets and so on.

To simplify the specification of protected vs. unprotected resources in the Tivoli Access Manager Policy Server, it is generally desirable to group resources under a common URL prefix, and protect or unprotect the contained resources as a group. For example:

<table>
<thead>
<tr>
<th>Command</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>protect</td>
<td>/secure</td>
</tr>
<tr>
<td>unprotect</td>
<td>/public</td>
</tr>
</tbody>
</table>

These rules can then be easily added to the Tivoli Access Manager Policy Server, and enforced by Tivoli Access Manager WebSEAL.

When an HTTP request arrives, WebSEAL works with the Policy Server to determine if the requested resource is protected. If it is protected, WebSEAL ensures that the user is authenticated before allowing the request to reach the application - for example, is the user signed up for the service? Did they specify the correct password?

WebSEAL is also used to establish a tenant context for the request. The tenant context includes metadata to identify which tenant the user belongs to so that appropriate restrictions can be enforced. For example, tenant 1 might sign up for SaaS application 1 and 2, while tenant 2 signed up for SaaS application 2 and 3. The tenant context is propagated downstream as HTTP request headers and informs downstream servers of the user identity, role, associated tenant identity and entitlements. The Apache HTTP server enforces coarse-grained entitlement by dispatching the HTTP request to the appropriate service based on the customer entitlement, provides support for load balancing and stickyness.

**Fine-grained access control**

An application may need to impose additional, finer-grained controls that further restrict access to protected resources. These additional controls may be based on user roles and/or tenant entitlements that are obtained from the tenant context. For example, a particular user may be entitled to create and edit a business process, but not entitled to deploy the process. A particular user may be assigned an administrative role so that the user can delete assets created by other members from the same account.

**Unique login and landing page for each SaaS application**

WebSEAL allows its various authentication-related pages to be customized using template pages. However for a multi-tenant SaaS delivery platform, the level of customization provided by WebSeal is not sufficient. What if we want different login page for each SaaS application or perhaps even tenant specific login page? Luckily we were able to find a solution using simple HTTP redirects. Figure 2 shows the overall approach.
Figure 2: Customized login page for each SaaS application backed by WebSEAL

A user navigates to the SaaS application login page (arrow 1). The SaaS application login page then uses XMLHttpRequest (XHR) to post credentials to the WebSEAL Login Service (arrow 2). The response (arrow 3) is parsed by the JavaScript in the SaaS application login page to detect success or login failure.

If a user attempts to directly access a protected page without first logging in (for example, by using a bookmark or following a link), the WebSEAL server will return its default login page. This default login page can be customized to redirect the user to the correct SaaS application login page.

This general approach can be extended to support additional WebSEAL pages to perform appropriate indirect - e.g. password reset; redirecting to different pages personalized for a user after successful login etc.

WebSEAL clusters for different access

One thing to note is that a WebSEAL server can be configured to either support basic access authentication or form-based authentication, but not both. Therefore, we have to deploy multiple WebSEAL servers, some configured for basic access authentication, and some configured for form-based authentication in order to support different kinds of client access such as Web forms, mobile access, Eclipse etc.

These WebSEAL servers share the common Tivoli Access Manager Policy Server, and are fronted by an Apache front-end reverse proxy that routes traffic to the correct WebSEAL server by using the Apache `mod_rewrite` module:

- If the request has a basic access header (that is: HTTP:Authorization), it forwards the request to the basic access enabled WebSEAL.
- Otherwise, it forwards the request to the form-based enabled WebSEAL.

The front-end reverse proxy also handles load balancing between the WebSEAL servers.
Using firewalls to enhance security

The IBM Cloud is a public cloud, this means that every virtual machine by default is reachable from anywhere in the world. However, we do not want all our virtual machines to be directly assessable; for example, the virtual machine that runs our database server should only be accessible by our application servers; the asset repository that we described in previous article should only be accessible by our SaaS applications, and so on. Firewalls play a key role for adding these extra layers of security protection.

In the Get started with the IBM SmartCloud Enterprise article (see Related topics), the authors explain the pros and cons for configuring firewall rules at the hypervisor level vs. at the virtual machines itself. Given the dynamic nature of our environment, where we need to easily add or remove virtual machines based on usage, we used the iptables approach to define firewall rules that are appropriate for each virtual machine.

A simple scenario for firewall rules

The base OS image only comes with one port open for input: port 22 so that you can SSH (Secure Shell) into it. However, when you install an application on a virtual machine, you want to open extra ports so your application can be accessed. For example, in Figure 3, there are two applications; application 1 access is via HTTP port 8081 and application 2 access is through HTTP port 8082.

Figure 3: Firewall rules example

However, you probably do not want to just open up these ports for anyone in the world to access without going through the proper access control. This is where you can add additional firewall rules for extra protection.

On virtual machine (9.8.7.6) where Application 1 is running, add a firewall rule such as:

```
-A INPUT -s 1.2.3.4 -p tcp -m tcp -dport 8081 -j ACCEPT
```

This opens port 8081 but it is only accessible by the virtual machine 1.2.3.4 which is the Apache Server.

Likewise, on virtual machine (5.1.2.3) where Application 2 is running, add a firewall rule to open port 8082 as follows:

```
-A INPUT -s 1.2.3.4 -p tcp -m tcp -dport 8082 -j ACCEPT
```

Now assuming the Apache Server (1.2.3.4) can be accessed via port 80, you can add the following firewall rule to the Apache Server to open only port 80 as follows:
Another common useful pattern is to enable a virtual machine to accept input from itself by enabling loop-back support. For example, you may want to deploy an operating system agent on a virtual machine so that it can report its status to the IBM Tivoli Enterprise Portal, the agent in this case, may need to connect to locally running services. To enable such loop-back support, add the following firewall rule:

```
-A INPUT -i lo -j ACCEPT
```

Finally, it is important to test your firewall rules and save them properly. Ensure the iptables is initialized with your new configurations when the virtual machine reboots and not with the default setup. You can issue the following commands to save the firewall rules:

```
/sbin/service iptables save
```

By carefully managing who can access what using authentication and firewall rules, you can secure your virtual machines that make up your cloud solution running in the public Internet.

**Summary**

Security is often cited as one of the primary inhibitors to cloud adoption. In this article, we showed you how to use matured products such as IBM Tivoli Access Manager for e-business to implement an access control solution that can protect your cloud resources and support a multi-tenant architecture. IBM has also published a red paper: "Cloud Security Guidance IBM Recommendations for the Implementation of Cloud Security" (see [Related topics](#)). It provides additional insights on what you should consider to have a secure cloud solution.
Related topics

- Tivoli Access Manager for e-business products page.
- IBM Tivoli Access Manager for e-business Redbooks paper.
- iptables wiki.
- In the developerWorks cloud developer resources, discover and share knowledge and experience of application and services developers building their projects for cloud deployment.

© Copyright IBM Corporation 2011
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)