Protect your SOA investments with Oracle Web Services Manager: Be Secure and Productive

By Peter Lorenzen, Logica

Traditionally, security was built into every single Web Service and open standards were not followed or did not exist. The result has been high costs for both maintenance and administration. Oracle Web Services Manager (OWSM) is a security product that solves these problems as well as others. Logica Denmark has done a larger SOA-based project for the Danish National Police where all Web Services, both internal and external accessible, are protected by OWSM. We also have a couple of smaller projects that use OWSM. In this article I will introduce the different components of the product and discuss our experience with it.

OVERVIEW

OWSM is an extra layer that encapsulates your Web Services. It is non-intrusive so there is no need to change existing Web Services. OWSM is built on open standards like the WS-Security standards family and provides a single interface to monitor and administer all your services real-time. The definition of security policies is done declaratively without any programming but if your requirements are not fulfilled by the built-in policy steps you can easily add Java code in a custom step. OWSM is mostly an administrator and compliance tool and not a developer tool.

POLICY ENFORCEMENT POINTS

OWSM contains two different types of Policy Enforcement Points (PEPs), Gateways, and Agents. A PEP can be viewed as a Firewall that all traffic to and from a Web Service has to pass through. The PEP decides if a call to the Web Service can be allowed through or not. Viewing a PEP as a firewall is a simplification. Securing a Web Service can involve tasks like authentication, authorization, encryption, decryption, signature verification, etc. A PEP has an associated security policy. A policy is broken down into a number of steps. A step could be to decrypt part of the message the PEP has just received. In Figure 1, you can see some of the build-in policy steps.

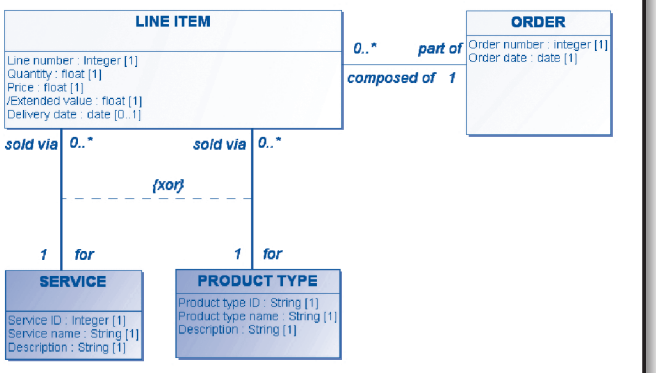

Figure 1 – Build-in Policy Steps

The build-in policy steps provide functionality that you can use to declaratively secure your Web Services. In most situations these build-in steps are all you need.

Figure 2 shows different use cases of Gateways and Agents and tries to visualize the communication taking place between the different OWSM components.

Gateways

A Gateway is a proxy, in addition to working as a kind of a firewall. When you register your Web Service with a Gateway, you get a new service URL and a new service WSDL URL, which you must always use when accessing the service. For example:

http://logica.dk:80/gateway/services/SID0003001
http://logica.dk:80/gateway/services/SID0003001?WSDL

The virtualisation of the Web Service URL also means the Gateway is ready for load balancing.

A Gateway provides:

- Transport protocol translation. Eg. http to JMS or MQ
- Content-based routing
- Failover to another URL if the service is down

From an administration and manageability point of view it makes sense to keep security away from the Web Service. The Web Service should solve a business requirement and not concern itself with anything else. As the Gateway will often be installed on a different server than the one where the Web Service is running, there can be an issue with security between the two. Information will be passed along in clear text. This is sometimes referred to as last mile security. If you have a requirement where this is a problem, you can use an Agent.

Agents

An Agent can be running either at the Web Service end, or at the Web Service consumer end. It is a more lightweight component that is running in the same memory space as the Web Service or the consumer. There is no problem with last-mile security since the Agent is running in the same Java container as the service. The following Java containers are supported: OC4J, Weblogic, AXIS, Websphere, TIBCO-BW. If the Web Service consumer does not support some of the security requirements that the Web Service provider demands, you can use an Agent at the consumer end.

I think most people will never need Agents but they can be used for last-mile security if no other solution is possible.

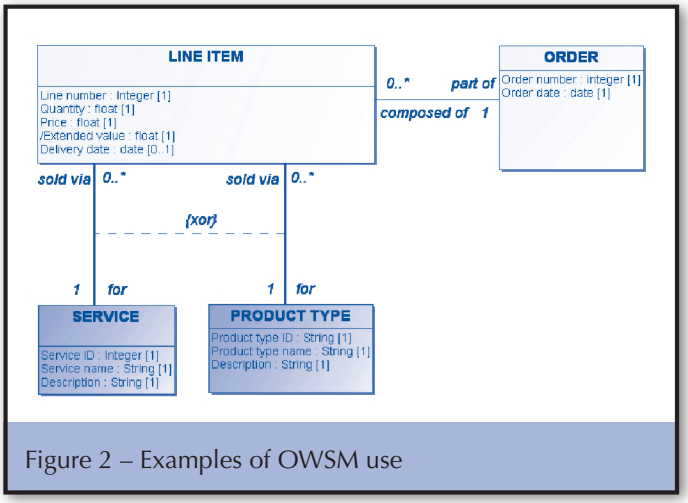

Figure 2 – Examples of OWSM use

THE OTHER OWSM COMPONENTS

In addition to Gateways and Agents, OWSM consists of the OWSM Control, the OWSM Policy Manager, and the OWSM Monitor. They are all running in a J2EE container on the application server. I will go into a bit more detail on each, but OWSM also consists of a couple of other "components". OWSM relies on a database to store information such as:

- Policies
- Monitoring data
- Message logs
- Service Level Agreement (SLA) data
- User defined Alarms
- Users and groups (only for standalone installations)

OWSM does not contain any tools to help manage and access certificates and encryption keys. All of these have to be accessed from the file system. OWSM supports Java Key Store and PKCS12 for trust stores. Oracle Wallet is also supported.

The OWSM Control

The way you administer and monitor OWSM is through the OWSM Control. It's an Enterprise Manager plug in and it's easy to use when you get to know it. All the products in Oracles SOA Suite have their own Enterprise Manager plug-in that looks nothing like the others, so it can be a bit difficult to navigate in the beginning.

The OWSM Policy Manager

The OWSM Policy Manager is a background process that handles the setup and administration of policies and steps in cooperation with the OWSM Control. It also communicates with PEPs regarding policies, etc.

When you define a policy for a Web Service in the current release of OWSM it is divided into four parts called pipelines: PreRequest, Request, Response, PostResponse. The Pre and Post pipelines are deprecated and will be removed in the next release so don't use them. The Request pipeline contains the steps that should be executed when a PEP receives a request to access a Web Service and the Response pipeline contains the steps that should be executed when a response is returned from the Web Service.

If you have many Web Services to protect, you can set-up templates for your Request and Response pipelines so you can be more productive. This is a good feature but unfortunately templates can only contain the build-in steps and not custom steps, which is a shame.

When you finish editing a policy for a Web Service and save the changes (It's actually called Committing) a new version of the policy is created. You can access all the old versions and even revert to an old policy if you want to. If you have a very dynamic system, remember to purge the old policy versions once in a while.

The OWSM Monitor

The OWSM monitor is a background process that collects and aggregates statistics. The statistic is available as different reports in the OWSM Control. The reports are HTML or Adobe Flash based, so they look nice on the screen but are difficult to print. Besides the built-in reports you can also define your own called Custom Views.

There are different kinds of statistics:

- Statistics/Monitoring
  - o Stored in the database
  - o Default automatically purged every 100 minutes. (max. is 60 days)
  - o Eg. Overall statistics, Security statistics, Service statistics
- Message Logs
  - o Stored in the database or in the database plus in a file
  - o Has to be manually purged
- Diagnostic logs from the different Java processes
  - o Stored in files

Notice that all the statistics are logged in the database are of a transient nature. Even though you have to manually purge the message logs, I would not like to keep years of data there since this was clearly not intended. If you, like we, have a requirement to store access logs for five years you have a problem. The log data is stored in a BLOB which doesn't make it easier to get your hands on it. We ended up creating a custom step for logging where we write to a central logging service. This solution was OK for our requirement but you cannot get all log data this way.

In addition to reports, you can also actively use the statistics logged in the database for alarms and SLA metrics. You can define alarms with a set of rules that trigger on a range of statistics, also aggregated statistics. You can also define which days of the week the alarm is active or at which hours of the day. When an alarm is triggered you can be notified by E-mail, SNMP, a call to a Web Service, or a call to a custom Java class. Alarms are good and versatile.

You can define SLA metrics like downtime and latency and get a report that document if you meet them. The metrics you can define are rather limited. There is no way of exporting the SLA statistics and the statistics are only there for 100 minutes up to 60 days, so it can only be used to show the current SLA status, not for the full history.

CUSTOM STEPS

Currently there are twenty-four build-in steps you can use for your policies. If your requirements are not met by these, you can develop your own in Java.

Many of the build-in steps have pre-requisite steps. For example, if you want to use the LDAP Authorize step you must first use a LDAP Authenticate step. We have a case where the authentication is done long before any Web Service is called. A SAML token is sent along in the request to the Web Service containing the user name but not the user's password. We must check if the user is allowed to access the Web Service by asking a LDAP server, e.g. a LDAP Authorize step. The only way to do this is via a custom step. First, we were a bit annoyed that you could not do a simple thing like a LDAP lookup without any prerequisites, but after we learned how simple it is to make a custom step, it was not a big issue.

SCALABILITY AND LICENSING

OWSM can scale horizontally by installing more Gateways on different servers. Most of the load is on the Gateway but it is also possible to have more than one Policy Manager. Do note that an extra Gateway also means an extra full OWSM license. Agents, on the other hand are free.

We have a customer that installed a Gateway on a server in their DMZ and the rest of the OWSM components on a server on their LAN. Even though they only have one instance running of each of the OWSM components they ended up paying for two full OWSM licenses. Keep this in mind when budgeting for an OWSM installation.

An OWSM license does not include a database license. The database has to be a version 9.2.0.7 or higher. You can use an Oracle Express Edition database or even a Lite database but I would advise against using any of these. If you take security serious enough to get OWSM, you should also use a real and fully supported database that you can get patches for.

OWSM AT THE WEB SERVICE CONSUMER

OWSM is not only a product for the Web Service provider. It can also be put to good use at the Web Service consumer end. An organization can use OWSM as a way to virtualize or externalize all the Web Services it consumes. This will result in easier administration and higher productivity. You can separate business logic and security. It is possible to define failover services if any exist and you can shift service providers without changes to your clients. Things like key and certificate administration can be centralized.

CONCLUSION

I believe OWSM is as much a productivity tool as a security tool. We have in the past been struggling with Web Service security in some of our Java Open Source projects. We spent a lot of time on something that can now be done very easily by using OWSM.

Our overall experience with OWSM has been very positive.

I had a quick look at my Metalink Service Request (SR) history. I have created 6 OWSM SRs and none of them ended up being a bug and mostly was requested for information or clarification. So OWSM is working and stable, but this leads to our biggest issue with OWSM – Information. OWSM is easy to use when you know how it is working, but it is very difficult to figure out how certain settings have to be used. There is often not much help to get from the online help and the documentation. But if you start by reading Sitaraman Lakshminarayanan's book and Vikas Jain's blog you have a good starting point. (See the OWSM resources in figure 3).

- Oracle Wiki http://tinyurl.com/6l3l9y
- Vikas Jain's Web Services Security Blog http://ws-security.blogspot.com/
- Sitaraman Lakshminarayanan's book http://tinyurl.com/58s2sm

Figure 3 – OWSM resources

Oracle Development Tools Member

About The Author

Peter Lorenzen is a technology manager at Logica Denmark. He has presented both at the ODTUG Kaleidoscope and the UKOUG conferences. He has published technical articles in the ODTUG Technical Journal *and the Danish* OracleEkspert *magazine and is a 10g DBA OCP.*