

BMC Software Inc.

Technical Disclosure Publication Document

Authors

Abhijit Rajwade
Ajoy Kumar

Secure Eventing-based Notification Mechanism for Business Data

Posted: July, 2009

Overview

This document describes a secure message notification mechanism (Publish-Subscribe model) for business data that imposes *per Message access control* (row and column level access control) and imposes *Authentication* as well as *Topic authorization* in order to provide a highly-secure messaging mechanism for any form of business data in a typical business scenario.

Background

Providing a secure message notification mechanism in a typical Publish-Subscribe model is a complex problem. The most important issue to address for solving this problem is to make sure that messages are delivered to the right subscribers who have the necessary authorization to receive the messages.

[JMS](#) (Java Messaging Service) is a popular open source standard for messaging. JMS is a part of the [J2EE](#) stack of technologies and enables the use of the J2EE security model with JMS. JMS includes optional inbuilt security features such as authentication, which several JMS vendors provide. When JMS is used outside of the J2EE stack or where there is a need to implement an application-specific security model, implementing secure message notification becomes very complex. Applications have their own security models where data access to users is restricted by using well-known techniques such as row level and column level security mechanisms commonly found in relational databases and SaaS (Software as a Service). When these applications publish messages using JMS, it is expected that the same security models be also applied and enforced for message access by users – this is particularly important in a Publish-Subscribe model and for multi-tenancy scenarios. This document presents a technique to conceptualize and design such a secure message notification system.

Copyright – BMC Software Inc.

Solution

In the publish/subscribe paradigm of message communication (such as JMS), a publisher publishes messages to a topic while multiple subscribers receive messages according to their subscriptions (based on which topics a subscriber subscribes to and what filters on the message content each subscriber sets up).

A publisher application needs to impose a security model with its own authentication and authorization schemes on users, topics and messages. For secure message notifications, each application develops pluggable security modules for authentication (user authentication) and authorization (topic level authorization and per message authorization) that can be installed and integrated on the JMS broker using a plug-in/interceptor capability on the JMS broker. These pluggable modules are the gatekeepers to effectively enforce the security as shown in the diagram below.

For example, in the diagram 1, there are two publisher applications A and B each having its own security models. A-Authe is an authentication module that verifies the identity of the user to access application A. A-TAutho is a topic authorization module that allows the user to subscribe to a “topic” belonging to A, and to receive only specific messages that the user is allowed to receive. When a user connects to the broker and subscribes to application A’s topic, then A-Authe and A-TAutho modules executing as part of the JMS broker will ensure that the user is authenticated and allowed to subscribe to the topic. Application A publishes messages to the JMS broker system with enough security information in the message to facilitate implementing the message authorization security model as a JMS-pluggable module, say A-MAutho and B-MAutho. When the JMS broker receives the message from application A, it will use the A-MAutho module to determine whether this message can be delivered to user A. If the A-MAutho module allows the delivery of this message, then this message is sent to the user, otherwise it is not. Hence, there could be multiple users subscribed to one topic, but some receive certain messages while others do not, based on the permission levels for each user. The message has a self-containing security information header for implementing the per message authorization. The security information in the message header is encrypted using any of the standard encryption techniques (for example one can use encryption techniques supported by the Java Cryptography Extensions – [JCE](#)) in order to secure this information when it is exchanged between the Publisher and the JMS broker. The security information in the message header is stripped from the message by the message-level authorization module before it is delivered to the Subscriber(s).

The proposed solution will work with the application’s security model. For example, if the application has row level (mechanism to determine which row is accessible to which user(s) based on their user credentials and/or group memberships) or column level (mechanism to determine which column is accessible to which user(s) based on their user credentials and/or group memberships) security for user data access, the same can be enforced by the JMS broker using these pluggable modules. The implementation details of how the modules implement the security model in JMS in a high performance

environment are outside the scope of this paper; they could, for example, communicate back with their parent applications, cache certain authentication information, use an encrypted security header in the message, etc. The publish/subscribe vendor – that is the JMS vendor must also allow a plug-in architecture where applications can install such modules. Viz. message interceptors are supported by Apache ActiveMQ.

For multi-tenant applications, the row-level security model (mechanism to determine which row is accessible to which user(s) based on their user credentials and/or group memberships) for data rows can be used to implement multitenancy. When the same application publishes messages in a multi-tenant scenario, it requires an encrypted row-level security header on all messages so that each subscriber receives messages that (s)he is authorized to view. Similarly, a multi-tenant application also requires that subscribers of one tenant only see messages for them and not from other tenants hosted on the same application server. This is accomplished by the per message authorization.

In a different embodiment of this invention (see diagram 2), the per-message authorization can be implemented as a layer in the application itself instead of implementing it as a pluggable module in the JMS broker. This approach will require per user or per group topics. From a scalability perspective, a topic per group is desirable as compared to a topic per user.

There is a clear business need for such a message-level security in publish/subscribe model. This approach provides a secure publish / subscribe messaging mechanism for business data, which imposes message level authorization, such as row and column level access control and imposes authentication as well as topic authorization in order to provide a highly- secure messaging mechanism in a typical business scenario. The invention essentially proposes a seamless integration of standard security mechanisms (authentication and authorization) with an application-specific / domain-specific security model for providing a highly-secure publish-subscribe mechanism.

**Drawings
Diagram 1**

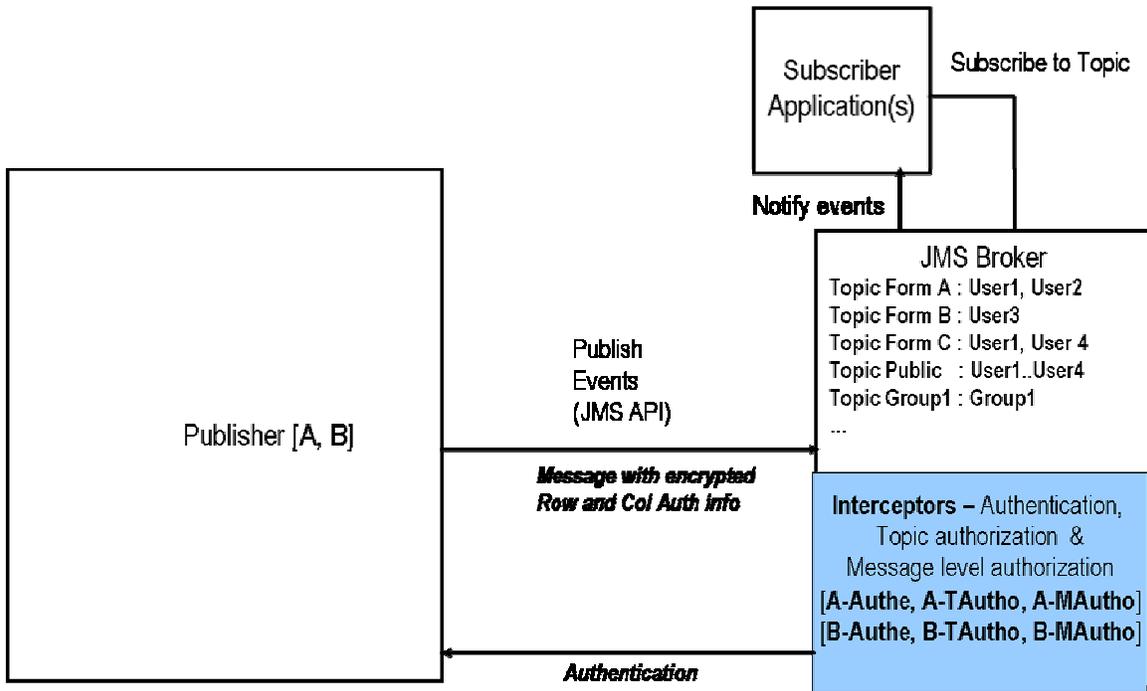


Diagram 2

