**🡪Securing JAX-RS**

I) Introduction

|-Types of Security Levels/mechanisms

|-1) Authentication

|-2) Authorization

|-3) Data protection (Encryption)

|-4) Integrity (Digital Signature)

|-5) Transport Level Security (TLS)

|-Authentication

|-1) HTTP Basic Authentication

|-Working Mechanism

|-Problem with this Approach and Solution

|-2) Form Based Authentication

|-3) HTTP Digest Authentication

|-Working Mechanism

|-Advantage

|-Disadvantage

|-4) HTTPS Client Authentication

|-Working Mechanism

|-Advantage

|-Disadvantage

|-Authorization

|-Encryption

II) Working Authentication, Authorization & Encryption in JAX-RS

|-Ways of Securing our JAX-RS

|-1. Configuration Approach

|-2. Annotation Driven Approach

|-3. Programmatic Approach

3) Summary

**1) Introduction:**

Many RESTful web services will want secure access to data and functionality that we wanted to provide. This is especially true for services that will be performing updates. They will want to prevent sniffers on the network from reading their messages. They may also want to fine-tune which users are allowed to interact with a specific service and disallow certain actions for specific users. The Web and the umbrella specification for JAX-RS, Java EE, provide a core set of security services and protocols that you can leverage from within our RESTful web services. These include:

Types of Security Levels/mechanisms:

1) Authentication

2) Authorization

3) Data protection (Encryption)

4) Integrity (Digital Signature)

5) Transport Level Security (TLS)

To work with 3, 4 we can use java provided API and 5 is can did on the server side.

We can impose JEE container security levels like Authentication, Authorization bcz these are coming from the HTTP protocol so we can apply these to our RESTful as well. But we cannot apply any enterprise level security policies like WSS-policies or SAML to RESTful services bcz these are specific to SOAP-Based services only.

Hence we can apply only HTTP provided security mechanisms only.

So in order to work with security for JAX-RS we need to relay on the Java EE Containers bcz these has support Authentication and Authorization.

**1) Authentication:**

Authentication is about validating the identity of a client that is trying to access your services. It usually involves checking to see if the client has provided an existing user with valid credentials, such as a password. The Web has a few standardized protocols you can use for authentication. Java EE, specifically our servlet container, has facilities to understand and configure these Internet security authentication protocols.

When you want to enforce authentication for your RESTful web services, the first thing you have to do is decide which authentication protocol you want to use. Internet protocols for authentication vary in their complexity and their perceived reliability. In Java land, most servlet containers support the protocols of Basic authentication, Digest authentication, and authentication using X.509 certificates. Let’s look into how each of these protocols works.

1) Basic Authentication

2) Form Authentication

3) Digest Authentication

4) Client Certificate Authentication

**2) Authorization:**

Once a client is authenticated, it will want to interact with our RESTful web service. Authorization is about deciding whether or not a certain user is allowed to access and invoke on a specific URI. For example, you may want to allow write access (PUT/POST/DELETE operations) for one set of users and disallow it for others. Authorization is not part of any Internet protocol and is really the domain of our servlet container and Java EE.

While authentication is about establishing and verifying user identity, authorization is about permissions. Is my user allowed to perform the operation it is invoking? None of the standards-based Internet authorization protocols discussed so far deals with authorization. The server and application know the permissions for each user and do not need to share this information over a communication protocol. This is why authorization is the domain of the server and application.

JAX-RS relies on the servlet and Java EE specifications to define how authorization works. Authorization is performed in Java EE by associating one or more roles with a given user and then assigning permissions based on that role. While an example of a user might be "John", "David" or "Danish" these people’s roles are used to identify a group of users, for instance, "administrator", "manager" or "employee". We do not assign access control on a per user basis, but rather on a per-role basis.

**3) Encryption:**

When a client is interacting with a RESTful web service, it is possible for hostile individuals to intercept network packets and read requests and responses if your HTTP connection is not secure. Sensitive data should be protected with cryptographic services like SSL. The Web defines the HTTPS protocol to leverage SSL and encryption.

JAX-RS has a small programmatic API for interacting with servlet and Java EE security, but enabling security in a JAX-RS environment is usually can also be done by using configuration and applying annotation metadata as well.

So now we will discuss and focuses on various web protocols for authentication and how to configure our JAX-RS applications to use authentication, authorization, and encryption in following ways.

**Ways of Working with Security:**

1. Configuration Approach

2. Annotation Approach

3. Programmatic Approach

Among these approaches we will use Programmatic Approach only bcz Programmatic Boot-strapping of RESTful will works agnostic to the RESTful Implementation vendor hence we will use Programmatic Approach in securing our Resources.