Jacob Griggs

01/30/2025

CS-230

**Module 4 Journal**

Using REST API allows for a smoother working between the server and client. This relationship allows there to be a distributed workload as the client does not share resources with the server. The client sends request for data to the server that is then processed and returned. However, these systems have a few disadvantages. It allows for both sides to be attacked individually, and information can be intercepted in the middle of the transfer.

Even with these problems, This system has been widely adopted. There are often two separate approaches to this online data communication: SOAP and RESTful. SOAP is an actual protocol maintain by the W3C Consortium and RESTful is more of an architectural framework.

The RESTful API architecture contains several things. It has to have separate client ad servers. Each request is stateless – meaning each request contains all the necessary information and does not rely on previous transmissions. Caching is expected to be used for fast response. Uniform interface to allow predictable API behavior must allow for unique identification of resources and a common vocabulary for manipulating those resources. Any layers, e.g., caching security, etc., must be transparent to the client and server. The server may, optionally, return code for the client to run.

The principal object in security represents the user whose credentials have been supplied in the API request. The Authenticator class is responsible for the actual authentication of the username and password. The designed API uses role-based authentication and the Authorizer class, used here, performs the role matching and decides if a user is allowed to perform a certain action.

Dropwizard then uses theses classes to authenticate and authorize any user trying to make a request on the API. To invoke this behavior the @AUTH annotation is used in conjunction with the roles annotation, @RolesPermitted, to authorize only certain roles. This is in opposite to the @PermitAll annotation which allows all roles access.

This pattern and implementation across multiple client-types would be possible. Because the RESTful API is not a protocol it should be transparent to the client. In the case of whether or not the API could be implemented on XPOX or PS4, as long as those platforms could formulate the request and receive the response no changes would be needed on the server side.

However, the gameauth service would need some additional work. Usernames of guest, user, admin are of little use and therefore an interface to allow new usernames/accounts and their associated roles to be added. These actions, adding users, and then assigning roles also needs to be controlled. As a guest can’t be able to make an admin account for themselves. This might be done through an API but could lead to major security risks.