Jonathan Roland

WEB 420 RESTful APIs

Instructor: Prof. Krasso

12/11/2020

Discussion 9.1

**Explain the difference between authentication and authorization.**

Authentication is the practice of confirming user identities. Authorization is the body of rules governing access to resources and operations on a per-user basis.

Authentication practices include the use of unique usernames or ids, passwords, pins, keys, one-time use tokens, biometrics, or other means to establish that someone is who they say they are. An authenticated client can be a real person, a session, or an application. The intent is to ensure that those clients interacting with a system are truly those who have a valid authorization to do so.

Authorization is the set of enforced rules for what actions a validated, authentic user can take. This often boils down to the rights to create, read, update or delete a resource. That resource could be a file, an object member, a user account, a RESTful resource or any number of other resources.

Let us use a fictional example. In the Star Trek universe, I am sure there are many people who would love to explore the Enterprise (being such a famous and accomplished ship). Star Fleet would not be quite so open about allowing just anyone on board such a powerful vessel, nor would her Captain. To certify the identity of each new Ensign that reports for duty, a datapad with official orders is presented when they embark. This is their *authentication*, proof that they are who they say they are. Further certification of their identity could be performed by verifying their records with Star Fleet (the central certificate authority) or performing a medical scan. Once the Ensign is known to be an authentic crew member, their *authorizations* are defined by their rank and station. They can’t just stroll up to the bridge or into the engine room as they please. We know what their authorizations are because we know who they are. The same concepts hold true for security with our real-world systems.

**Select two HTTP status codes and describe how they are used and why they are important.**

501 – Not Implemented

This response relates to an unrecognized request method. If a client requests a method that the API does not support, the API can respond with 501 to inform the client that the request is ‘not implemented’ (aka does not exist *currently*). The wording leaves open the possibility that this could be a supported method in the future because the client may have made a good faith request to the wrong version of the API.

302 – Found

The requested resource resides temporarily under a different URI. The client can refer to the Request-URI header for future requests for this resource. The client should not cache this response unless specified in the Cache-Control or Expires headers. The temporary URI for the resource can be found in the Location field of the response. This response code is specifically used to encourage the client to pause and consider the relocated condition of the requested resource, rather than automatically redirect to it. The exception is if the request type is GET or HEAD.

**References:**

*Authentication vs. Authorization | Okta*. (n.d.). Okta, Inc. Retrieved December 12, 2020, from https://www.okta.com/identity-101/authentication-vs-authorization/

Todd Fredrich, Pearson eCollege. (n.d.). *HTTP Status Codes*. REST API Tutorial. Retrieved December 12, 2020, from https://www.restapitutorial.com/httpstatuscodes.html