Shibboleth and TIER at UMBC

UMBC's use of Shibboleth dates to the mid-2000s, when we ran Shibboleth Identity Provider version 1. Our first SAML integration went live circa 2007. We upgraded to IdP v2.0 (and SAML 2.0) in 2010, and IdP v3.0 in 2015.

UMBC has had some form of web single sign-on since 2000, when we launched a home-grown SSO service, called WebAuth, which functions similarly to CAS. Old habits die hard, and in fact, we're still running the WebAuth service today. Several important web applications continue to rely on it, and it handles front-end authentication for the IdP (via the external authentication plugin). Our long-term goal is to move off WebAuth, and use the Shibboleth IdP exclusively for both authentication and authorization. However, that is not going to happen in the immediate future, so for now, we need to find a way for WebAuth to coexist with the TIER version of the Shibboleth IdP. Currently, they both reside on the same server, with Apache running the front-end AuthN system and proxying requests to the IdP using mod_proxy_http.

Why go to TIER in the first place? Well, it will be a big win for us operationally. Our current setup consists of three VMs behind a load balancer, each running identical configurations of the IdP and WebAuth. The IdP administrator (me) handles operational aspects of the identity provider, including configuration, customization, and upgrades. A separate unit within our division handles lower-level system administration of the VMs themselves, including patching, backups, and security incident response. In general, this division of responsibilities works well; however, there's currently no mechanism in place for maintaining a consistent configuration across all three load-balanced nodes. Whenever I have to make a change (e.g. to add an attribute release rule, or load metadata for a new relying party) I have to manually propagate the change to each of the servers. It's tedious and error-prone, and leads to inconsistencies. For example, if one of the VMs is down at the time I make the change, and later comes back up, it will have an older version of the IdP configuration until I manually intervene. While the system administration group has methods in place to facilitate replication, I'm not up to speed on the system they use; and conversely, they're not familiar enough with the IdP to handle this on their end.

TIER, and the containerization model, promise to make things better for us. Having no real-world experience running Docker containers in production, we still have a significant learning curve ahead of us; however, I think switching from our existing system to a DevOps model will eventually pay dividends. Just to name a single example: replication will be a lot easier, as we'll have a single "master" copy of the IdP configuration that we'll use to generate as many running containers as we need, each behind a (yet-to-be-determined) load-balancing mechanism. Also, synchronization is no longer an issue, as older containers can just be spun down and replaced with new containers.

In my next entry, I'll go into more detail about how we plan to migrate from our existing IdP configuration to a TIER DevOps model.