Keystone

Keystone is the commonly used Identity provider in OpenStack. It may also be used for authentication in Object Storage. Coverage of securing Keystone is already provided in other documentation.

SWAuth

SWAuth is another popular alternative to Keystone. In contrast to Keystone it stores the user accounts, credentials, and metadata in object storage itself. More specifics about where the objects are stored can be found on the SWAuth website at http://gholt.github.io/swauth/.

SWAuth has these types of roles (or groups) for a user:

.super_admin	Can perform any action on any OpenStack Account, Container, or Object
.reseller_admin	Can perform most actions on any OpenStack Account, Container, or Object. Cannot create other reseller admins.
.admin	Can perform actions limited to the single OpenStack Account it belongs to
Regular User	Can access containers or objects they have permission to in the OpenStack Account to which they belong

The following table provides a matrix of what each role/group can do:

Figure 21.3. Object storage SWAuth role matrix

Role/Group	Capabilities											
	Get List of Accounts (Get Admin Info, get_reseller)	*Get Account Details (Users, etc.)*	*Create Account*	*Delete Account*	*Get User Details*	*Create User - Admin*	*Create User - Reseller Admin*	*Create User - Regular*	*Delete User*	*Set Service Endpoints*	*Get Account Groups*	*Modify User/Change User Password*
.super_admin (username)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
.reseller_admin (group)	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
.admin (group)	✗	✓	✗	✗	✓	✓	✗	✓	✓	✗	✓	✓
Regular user (type)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Warning

The super admin key is stored in `/etc/swift/proxy-server.conf` and MUST be protected! See the File

Permissions section for guidance on protecting this file. Frequent changing of this key is recommended.

One approach for administration is to create an OpenStack Object Storage Account called "CloudAdmins" and create reseller_admin users in that account. Each user will be able to do administrative functions in all the other accounts. Creating a reseller_admin will require the super admin key.

Another useful way to secure the super admin key is to have it exist only on the proxy server and retrieve the key on-demand via ssh or by running the command on the proxy server itself and using a grep to extract the key on the fly.

Protecting cloud administration

When using SWAuth you can actually designate that certain proxy service nodes are to NOT allow administrator API calls. This is useful if you have Proxy service nodes on the public Internet and wish to restrict administration functions to only special Proxy service nodes on a private network. This is done by setting the `allow_account_managment` to false in your `proxy-server.conf`.

Another important consideration is that the SWAuth command line tools expose the user credentials on the command-line. The system from which they are executed must be secure to prevent disclosure in the process list to other uses. Another option is to use the SWAuth admin REST API to implement your own admin CLI tools that don't expose the key as a command-line option.

Salting and hashing passwords

SWAuth by default stores passwords in clear-text. It also offers a sha1 hashing provider, but the salt used is global. Additionally, no iterations or key stretching is performed. This is a limitation of SWAuth.

You may optionally add-in your own hashing code or provider as a hook to SWAuth. See the SWAuth code and site for details.

If you use the global salt be sure to secure it and back it up. If you have multiple proxy nodes each one has to have a copy so that may be good enough for you. If you ever lose it or change it then all existing user passwords will not work and will have to be reset.

You should make sure the salt you choose is generated using a cryptographically secure random number generator and of sufficient length. At least 20 characters is recommended.

The salt is stored in the `/etc/swift/proxy-server.conf` file which must be secured with proper ACLs. See the File Permissions section for guidance.

Other notable items

In /etc/swift/swift.conf on every service node there is a "swift_hash_path_suffix" setting. This is provided to reduce the chance of hash collisions for objects being stored and avert one user overwriting the data of another user.

This value should be initially set with a cryptographically secure random number generator and consistent across all service nodes. Ensure that it is protected with proper ACLs and that you have a backup copy to avoid data loss.