Authentication and Authorization to Cloud Resources

Harvard Example

Core Scope and Requirements:

- Eliminate the use of AWS-native credentials for Authentication to AWS administrative tools for People, for both the console UI and in a command line interface
- Replace AWS-native credentials with existing HarvardKey credentials plus a second factor (Duo)

https://confluence.huit.harvard.edu/display/CLA/Common+Auth+Documentation

Emory Example

Requirements similar to Harvard's. We have 2 similar implementations, one for, primarily, researchers, and one for infrastructure (data center migration):

- AWS at Emory IDM Configuration
- Infrastructure Project IDM Configuration

Penn State Example

In AWS an account owner provisions users who will have access to the AWS console. Using AWS Identity and Access Management (IAM) roles with specific levels of permissions can be assigned to users by the account owner.

When provisioning an AWS account, a root account is created. This account has access to do the highest level of privileged tasks in the AWS console. It can delete the account, it can create IAM resources and link other AWS accounts for billing purposes. If used improperly, it can delete its own data centers. It is not advisable to use this account unless absolutely necessary. However, AWS offers several ways to integrate Identity and Access Management capabilities that allow console users to operate with more granular privileges. In fact, many different SAML integrations are supported that allow for a multi-user console environment where users operate with specific roles (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_saml_3rd-party.html).

An IAM role is similar to a user, it is associated with a permission policy that describes what is can and can not do in the AWS account. Roles can be a one to one or one to many association. It is easy to make simple rules that allow a role to do simple tasks like launch and terminate EC2 instances or read from an S3 bucket. But, it is also possible to make complex roles that limit a role to operate with very specific privileges in a certain part of the environment. For example, a role could be created for a developer team that would only allow these users to operate in a specific dev-test "data center" (Virtual Private Cloud or VPC). Or, a role could be created that would only be able to spin up certain size instances to ensure that spending doesn't get out of control. An infrastructure role could be created such that only this group could be able to perform typical infrastructure tasks like modifying subnets, VPCs or load balancers configuration.

Integrating AWS IAM roles with campus IdM is possible in many different scenarios. A common scenario might be the integration of a campus Shibboleth IDP with AWS. Each AWS account needs to have the IDP metadata configured in it. This can be accomplished manually through the console or via the AWS API. The AWS account will require the roles to be created in IAM with the level of privilege needed. This can be done manually through the console or via API. More information on using AWS IAM roles with Shibboleth can be found in this blog post: http://blogs.aws.amazon.com/security/post/TxRTTT5PLUE6B5/How-to-use-Shibboleth-for-single-sign-on-to-the-AWS-Management-Console.

In the simplest scenario, an IDP could be configured to pull group memberships for a user that is logging into AWS from an LDAP directory. One of the easiest ways to accomplish this is to develop a naming scheme for IAM roles that contain both Amazon Resource Names (the 12 digit number identifying the account) and a role.

The IDP could be configured to look for LDAP groups with a certain regular expression string and parse that information into the attributes that AWS requires (awsRoles and awsRoleSessionName). For example, inserting a person into a group named aws.123456789012.read-only could inform AWS that the user is logging into account 123456789012 and that the user should assume the IAM role with the name "read-only" in that account. If the user is in multiple groups and multiple attributes are asserted, the user has the choice of what account/role they wish to assume (see figure 1-1).

Grouper (http://www.internet2.edu/products-services/trust-identity-middleware/grouper/) is a popular enterprise access management system that is highly customizable and supports group management. It supports the use of naming stems, a string that forms the leading part of a Group's name. In this way, the possibility that different groups can be given the same name is substantially reduced, and the name of each group can be made to reflect something about the authority under which it was created. The permission on the stem can be configured such that only a particular application or process could modify it.

Example attribute resolver stanzas for group naming convention in the form of "aws.123456789012.read-only"

```
<resolver:AttributeDefinition id="awsRoles" xsi:type="ad:Mapped" sourceAttributeID="MemberOf">

 <resolver:Dependency ref="ldap"/>

 <resolver:AttributeEncoder

 xsi:type="enc:SAML2String"

 name="https://aws.amazon.com/SAML/Attributes/Role" friendlyName="Role" />

 <ad:ValueMap>

 <ad:ReturnValue>arn:aws:iam::$1:saml-provider/Shibboleth,arn:aws:iam::$1:role/$2</ad:ReturnValue>

 <ad:SourceValue>cn=aws.([^.]*).([^,]*),.*</ad:SourceValue>

 </ad:ValueMap>

</resolver:AttributeDefinition>

<resolver:AttributeDefinition id="awsRoleSessionName" xsi:type="ad:Simple" sourceAttributeID="mail">

 <resolver:Dependency ref="ldap"/>

 <resolver:AttributeEncoder

  xsi:type="enc:SAML2String"

  name="https://aws.amazon.com/SAML/Attributes/RoleSessionName"

  friendlyName="RoleSessionName" />

</resolver:AttributeDefinition>
```

A caveat with the use of roles is that roles do not support MFA. However, the institution could enforce MFA at the IDP or SSO layer. One of the obvious advantages of using IAM roles over IAM users is that there are no credentials on the AWS side to manage. When a user is taken out of the group or they leave the institution, they lose access to the AWS environment.

User is given login URL (in the case of AWS, it is an IDP initiated SSO handler in the form of https://idp.example.edu/idp/profile/SAML2/Unsolicited/SSO?providerId=urn:amazon:webservices)

University of Illinois - Urbana-Champaign Example

InCommon Case Study