How to Rotate Access Keys for IAM Users

by Ben Brauer | on 02 OCT 2013

Changing access keys (which consist of an access key ID and a secret access key) on a regular schedule is a well-known security best practice because it shortens the period an access key is active and therefore reduces the business impact if they are compromised. Having an established process that is run regularly also ensures the operational steps around key rotation are verified, so changing a key is never a scary step.

In an earlier [post](http://blogs.aws.amazon.com/security/post/Tx1XG3FX6VMU6O5/A-safer-way-to-distribute-AWS-credentials-to-EC2), we described Identity and Access Management (IAM) roles for Amazon EC2. If you run applications on EC2 that need access to AWS services, we strongly recommend using this feature. Roles use temporary security credentials that auto-expire and auto-renew, so you don’t have to worry about access key rotation – AWS does it for you. However, if you are running applications somewhere other than on EC2, you should add access key rotation to your application management process. In this post, Cristian Ilac, software development manager on the IAM team, will walk you through the steps to rotate access keys for an IAM user.

To rotate access keys, you should follow these steps:

1. Create a second access key in addition to the one in use.
2. Update all your applications to use the new access key and validate that the applications are working.
3. Change the state of the previous access key to inactive.
4. Validate that your applications are still working as expected.
5. Delete the inactive access key.

**Key Rotation Example**

Here’s an example of the key rotation steps listed above. You are an administrative IAM user and will use the [AWS Command Line Interface (CLI)](http://aws.amazon.com/cli/) to rotate access keys for a single user, Alice. The CLI uses IAM APIs, so the same steps can be performed programmatically via the [AWS SDK](http://aws.amazon.com/tools/), or using the web-based UI of the IAM Management Console like we showed in a previous [post](http://blogs.aws.amazon.com/security/post/Tx1R9KDN9ISZ0HF/Where-s-my-secret-access-key). Because keys are considered sensitive information, you should perform all of these commands only on a trusted computer.

After [installing](http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html) the CLI, run the following command to see what Alice’s access keys are:

aws iam list-access-keys --user-name Alice

The command returns something like this:

{

"AccessKeyMetadata": [

{

"UserName": "Alice",

"Status": "Active",

"CreateDate": "2013-04-03T18:49:57Z",

"AccessKeyId": "AKIAI44QH8DHBEXAMPLE"

}

]

}

**Step 1: Create a second access key**

Create a new (second) access key for Alice using this command:

aws iam create-access-key --user-name Alice

This returns:

{

"AccessKey": {

"UserName": "Alice",

"Status": "Active",

"CreateDate": "2013-09-06T17:09:10.384Z",

"SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",

"AccessKeyId": “AKIAIOSFODNN7EXAMPLE"

}

}

Notice that AWS IAM commands use unique access key identifiers (AKIDs) to refer to individual access keys. You can use the AKIDs to identify and manage the access keys your application uses. Key creation is the only time AWS will expose the secret associated with the access key in clear text. Record it and store it securely.

Now Alice has two active access keys. Note that AWS only allows for two keys per user.  If you already have two active access keys, you will not be able to create a third one.

Listing the keys using the list-access-keys command shows both keys:

aws iam list-access-keys --user-name Alice

{

"AccessKeyMetadata": [

{

"UserName": "Alice",

"Status": "Active",

"CreateDate": "2013-04-03T18:49:57Z",

"AccessKeyId": "AKIAI44QH8DHBEXAMPLE"

},

{

"UserName": "Alice",

"Status": "Active",

"CreateDate": "2013-09-06T17:09:10.384Z",

"AccessKeyId": "AKIAIOSFODNN7EXAMPLE"

}

]

}

**Step 2: Distribute your access key to all instances of your applications**

After creating the new key, you will distribute it and instruct your application to transition to using it.  Before moving on to the next step ensure that all instances of your application are indeed using it and that they function correctly.

**Step 3: Change the state of the previous access key to inactive.**

Disable the old access key using this command:

aws iam update-access-key --access-key-id AKIAI44QH8DHBEXAMPLE --status Inactive --user-name Alice

To verify that the key has been disabled, use this command to list the active and inactive keys for Alice:

aws iam list-access-keys --user-name Alice

You’ll see something like this:

{

"AccessKeyMetadata": [

{

"UserName": "Alice",

"Status": "Inactive",

"CreateDate": "2013-04-03T18:49:57Z",

"AccessKeyId": "AKIAI44QH8DHBEXAMPLE"

},

{

"UserName": "Alice",

"Status": "Active",

"CreateDate": "2013-09-06T17:09:10.384Z",

"AccessKeyId": "AKIAIOSFODNN7EXAMPLE"

}

]

}

**Step 4: Validate that your application is still working as expected**

Once the key has been marked inactive, it cannot be used for authenticating AWS service API calls; therefore, you should verify at this point that your application still works. That’s why we just disabled the old access key first: if something were to go wrong, you could quickly re-enable the previous access key using the aws iam update-access-key command.

**Step 5: Delete the inactive access key.**

The last step is deleting the inactive access key by using this command:

aws iam delete-access-key --access-key-id AKIAI44QH8DHBEXAMPLE --user-name Alice

Note that the deleting action – unlike disabling the access key – is an irreversible operation. After deletion completes, an access key is no longer available. You can list Alice’s access keys again to confirm that you removed the old access key:

aws iam list-access-keys --user-name Alice

The old access key is gone from the results:

{

"AccessKeyMetadata": [

{

"UserName": "Alice",

"Status": "Active",

"CreateDate": "2013-09-06T17:09:10.384Z",

"AccessKeyId": "AKIAIOSFODNN7EXAMPLE"

}

]

}

**Conclusion:**

If you’re using EC2 instances, we recommend using roles for EC2 in order to automatically rotate access keys.  If you are not using EC2, we suggest manually rotating keys on a periodic basis as a security best practice. For more information on access keys and rotation procedures, please visit the AWS IAM [documentation](http://docs.aws.amazon.com/IAM/latest/UserGuide/Using_RotatingCredentials.html).