**Automation of AWS IAM Key Rotation**

**Note:**

* Replace the value <iam-username> with ncif-hpcdm-svc for DDAS non-prod and with ncifhpcdmsvcp for DDAS prod.
* Replace the value <account-number> with 828870720727 for DDAS non-prod and with 537226181452 for DDAS prod.
* ~~To create ‘IAMLimitedAccess’ policy, for “Resource” replace the value <iam-username> with ncif-hpcdm-svc & <account-number> with 828870720727~~.
* To create ‘dme-iam-user-read-access-specific-secret’ policy, copy Secret ARN from the Secrets Manager previously created and paste it in the “Resource.”

1. **Create SNS topic and SNS email subscription that will be used in the SNS topic arn section in the python code for the Lambda Function**

* Navigate to the SNS section of the AWS console
* Click create topic > select standard
* Insert name (sns\_keyrotation) > click create topic
* Click create subscription
* Click protocol > select email
* Insert ‘HPC\_DME\_Admin@mail.nih.gov’ > click create subscription

1. **Create least privilege permission to SNS and IAM policy and add policies to the serverless role**
2. **Create an IAMLimitedAccess policy**

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"iam:DeleteAccessKey",

"iam:GetAccessKeyLastUsed",

"iam:UpdateAccessKey",

"iam:ChangePassword",

"iam:CreateAccessKey",

"iam:ListAccessKeys"

],

"Resource": "arn:aws:iam::<account-number>:user/<iam-username>"

},

{

"Sid": "VisualEditor1",

"Effect": "Allow",

"Action": [

"iam:GenerateCredentialReport",

"iam:UpdateAccountPasswordPolicy",

"iam:ListUsers"

],

"Resource": "\*"

}

]

}

1. **Create an SNSReadWriteAccess policy**

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"sns:Publish",

"sns:GetTopicAttributes",

"sns:CreateTopic",

"sns:SetTopicAttributes",

"sns:Subscribe",

"sns:ConfirmSubscription"

],

"Resource": "arn:aws:sns:us-east-1:<account-number>:sns\_keyrotation"

},

{

"Sid": "VisualEditor1",

"Effect": "Allow",

"Action": [

"sns:GetEndpointAttributes",

"sns:SetSubscriptionAttributes",

"sns:Unsubscribe",

"sns:GetSubscriptionAttributes"

],

"Resource": "\*"

}

]

}

1. **Create serverless role for the Lambda Function and attach permissions**

* Navigate to the IAM dashboard of the AWS Console
* Select 'Roles' from the 'Access management' menu on the left
* Click create role
* Select type of trusted entity (AWS service[default])
* Choose a use case Lambda
* Click next permission
* Filter policies > search and select policies (SNSReadWriteAccess, IAMLimitedAccess, SecretsManagerReadWrite & AWSLambdaBasicExecutionRole)
* Click next tags > click next review
* Give role name (e.g., serverless\_role)
* Create role

1. **Create the Lambda Function with the above serverless\_role**

* Navigate to the Lambda portion of the AWS Console
* Click create function
* Select **Author from scratch**
* Name the Lambda Function
* Select **Python3.6** as runtime
* Select change default execution role > choose an existing role (serverless\_role)
* Click Create function
* Copy the python code from the attached python code file and paste in the Lambda function
* Edit Python code in the Lambda Function. Copy <SNS ARN> from SNS topic created in No. 2 and paste into **<ops\_sns\_topic>** section
* Deploy Lambda function

1. **Store AccessKeyId and SecretAccessKey securely in the Secrets Manager and retrieve during an automated rotation deployment with Lambda function invocation.**

* Navigate to the AWS Secrets Manager section of the AWS console.
* Select store a new key
* Select secret type > Other type of secrets (e.g. API key)
* Choose Secret key/value – add AccessKeyId (key) and AccessKeyId (value). *Get info from IAM user credential*
* Select Add row to add Secret key/value – add SecretAccessKey (key) and SecretAccessKey (value). *Get info from IAM user credential*
* Click next
* Secret name (<iam-username>)
* Description (e.g. IAM User for DME S3 bucket}
* Click next
* Configure automatic rotation > choose disable automatic rotation (default)
* Click store
* Record ARN of stored secret (**<arn-of-stored-secret>**) for use in next step

1. **Create and attach IAM policy “dme-iam-user-read-access-specific-secret” policy to IAM user <iam-username> for read-only access to the specific Secrets Manager for the key rotation.**

**i.) Create IAM policy dme-iam-user-read-access-specific-secret**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"secretsmanager:GetResourcePolicy",

"secretsmanager:GetSecretValue",

"secretsmanager:DescribeSecret",

"secretsmanager:ListSecretVersionIds"

],

"Resource": "**<arn-of-stored-secret>**"

}

]

}

1. **Create CloudWatch event rule to invoke lambda function for Access key rotation**

Create the CloudWatch Events rule that will rotate AccessKeyId and SecretAccessKey in 120 days and initiates an email via SNS.

1. Create **“create event”**

* Navigate to CloudWatch > Events > Rules
* Click Create rule at the top of the console
* Select schedule
* Select cron expression {0 12 1 4,8,12 ? \*}
* On the right side, click add targets
* Select Lambda > select function (serverless\_role)
* Configure input > select Constant {"action":"create","user":"<iam-username>"}
* Click Configure details
* Configure the rule with a name (CreateKey) and save the rule

1. Create **“deactivate event”**

* Navigate to CloudWatch > Events > Rules
* Click Create rule at the top of the console
* Select schedule
* Select cron expression {0 12 1 5,9,1 ? \*}
* On the right side, click add targets
* Select Lambda > select function (serverless\_role)
* Configure input > select Constant {"action":"deactivate","user":"<iam-username>"}
* Click Configure details
* Configure the rule with a name (DeactivateKey**)** and save the rule

1. Create **“delete event”**

* Navigate to CloudWatch > Events > Rules
* Click Create rule at the top of the console
* Select schedule
* Select cron expression {0 12 1 6,10,2 ? \*}
* On the right side, click add targets
* Select Lambda > select function (serverless\_role)
* Configure input > select Constant {"action":"delete","user":"<iam-username>"}
* Click Configure details
* Configure the rule with a name (DeleteKey) and save the rule