Access Keys for AWS CLI Using Cornell Two-Step Login - Shibboleth

Use Case

This document shows how to setup and use the awscli-login tool to retrieve temporary AWS access keys using your Cornell netid credentials and Duo (i.e., Cornell Two-Step Login). Using temporary access keys associated with an AWS role to authenticate to the AWS Command Line Interface (CLI) is much safer than using fixed AWS access keys tied to an IAM user. Now that this option is available to Cornell AWS users, we recommend that fixed access keys no longer be used for humans using the AWS CLI.

Prerequisites

The awscli-login tool does NOT work with AWS CLI version 2.x, however, AWS SSO does and is available now! Check out our docs and send us a ticket to request access to AWS SSO.

We are told that awscli-login now works with Windows, though we don't yet have any Windows-specific instructions here. Let us know if you can provide some!

- If you don't have the AWS CLI v1 installed yet, that's great. Start by checking that you have Python 3.5+, then worry about the AWS CLI.
  - AWS CLI installation help: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
  - Python installation information: https://wiki.python.org/moin/BeginnersGuide/Download (for non-beginners: follow links on the page as appropriate)
- If you already have the AWS CLI (v1) installed, you'll need to make sure that it is using Python 3.5+. If it isn't, then the awscli-login plugin won't work properly and installing it may break AWS CLI installs that are NOT using Python 3.5+.
- Docker with the awscli login tool with other helpful cloud utilities are available in this repo https://github.com/CU-CommunityApps/ct-cloud-utils-dockerized

Linux/Mac OS

```
$ python --version
Python 2.7.14
$ python3 --version
-bash: python3: command not found
$ aws --version
aws-cli/1.15.83 Python/2.7.14 Linux/4.14.77-70.59.amzn1.x86_64 botocore/1.10.82
```

In the above example, Python 2.x is installed and the AWS CLI v1 is installed, but using Python 2.x. That's exactly what we don't want. It is beyond the current scope of this article to describe how to install Python 3.5+, and the AWS CLI, ensuring that the CLI is using your Python 3.5+ installation. Please note that virtualenv may make it easier to get the AWS CLI installed and using Python 3.5+.

Install and Configure awscli-login

Don't bother trying to install or use awscli-login until your installed version of the AWS CLI v1 reports that it is using Python 3.5+. You will also need to ensure that `pip` is using Python 3.5+. That may mean you will need to use `pip3` in the commands below, instead of plain `pip`.

```
$ pip install --upgrade awscli-login
...
$ aws configure set plugins.login awscli_login
...
$ aws login configure
ECP Endpoint URL [None]: https://shibidp.cit.cornell.edu/idp/profile/SAML2/SOAP/ECP
Username [None]: <YOUR NETID>
Enable Keyring [False]:
Duo Factor [None]:
Role ARN [None]:
$
```

Using awscli-login

Login using defaults setup above:

```
$ aws login
Password: ********
Factor: push
# Provided second factor out of band
Please choose the role you would like to assume:
     Account: 000011112222
         [ 0 ]: shib-admin
     Account: 777788889999
         [ 1 ]: shib-admin
         [ 2 ]: shib-cs
         [ 3 ]: shib-dba
Selection: 1
$ aws sts get-caller-identity
{
    "UserId": "AROAICCPMY7VALLFYHWPA:pea1@cornell.edu",
    "Account": "777788889999",
    "Arn": "arn:aws:sts::777788889999:assumed-role/shib-admin/pea1@cornell.edu"
}
```

The options for "Factor" are "push", "sms", "phone", "auto"

Advanced Use

```
$ aws --profile foo login configure
ECP Endpoint URL [None]: https://shibidp.cit.cornell.edu/idp/profile/SAML2/SOAP/ECP
Username [None]: pea1
Enable Keyring [False]:
Duo Factor [None]: auto
Role ARN [None]: arn:aws:iam::111111111111:role/shib-admin

$ aws --profile bar login configure
ECP Endpoint URL [None]: https://shibidp.cit.cornell.edu/idp/profile/SAML2/SOAP/ECP
Username [None]: pea1
Enable Keyring [False]:
Duo Factor [None]: auto
Role ARN [None]: arn:aws:iam::222222222222:role/shib-admin

$ aws --profile foo login
Password: **************
# Provided second factor out of band
$ aws --profile foo sts get-caller-identity
{
    "Arn": "arn:aws:sts::111111111111:assumed-role/shib-admin/pea1@cornell.edu",
    "Account": "111111111111",
    "UserId": "XXXXICCPMY7VALLFXXXX:pea1@cornell.edu"
}


$ aws --profile bar login
Password: **************
# Provided second factor out of band
$ aws --profile bar sts get-caller-identity
{
    "Arn": "arn:aws:sts::222222222222:assumed-role/shib-admin/pea1@cornell.edu",
    "Account": "222222222222",
    "UserId": "YYYYICCPMY7VALLFYYYY:pea1@cornell.edu"
}
```

Troubleshooting

Help! I got a 401 Client Error

```
$ aws login
Password: **************
401 Client Error: Unauthorized for url: https://shibidp.cit.cornell.edu/idp/profile/SAML2/SOAP/ECP
```

This situation occurs when you provide an incorrect password.

> If you have setup your awscli-login config to save your password (i.e., enable_keyring = true) then the plugin will happily save an incorrect password in the keyring. In this situation you will never be prompted for a password and you will immediately get a 401 error. To work through this edit your ~/.aws-login/config file to set enable_keyring = false, to be prompted for a password again.

Help! I got a 504 Server Error

```
$ aws login
Password: **************
504 Server Error: GATEWAY_TIMEOUT for url: https://shibidp.cit.cornell.edu/idp/profile/SAML2/SOAP/ECP
```

This situation occurs when you fail to provide your Duo second factor in time.